



A11104 260288

NIST  
PUBLICATIONS

**NISTIR 5394**

**VOL II OF II - APPENDICES**

# **The Design and Development of An Information Retrieval System for the EAMATE Data**

**Natalie Willman  
Laura L. Downey**

U.S. DEPARTMENT OF COMMERCE  
Technology Administration  
National Institute of Standards  
and Technology  
Gaithersburg, MD 20899

*Welcome*

TO THE EAMATE PROTOTYPE



DESIGNED AND DEVELOPED BY

**Natalie E. Willman,  
Senior Computer Scientist**

**Laura L. Downey,  
Computer Scientist**

**National Institute of Standards  
and Technology**

Continue

QC  
100  
.U56  
1994  
#5394

**NIST**







# **The Design and Development of An Information Retrieval System for the EAMATE Data**

**Natalie Willman  
Laura L. Downey**

U.S. DEPARTMENT OF COMMERCE  
Technology Administration  
National Institute of Standards  
and Technology  
Gaithersburg, MD 20899

April 1994



**U.S. DEPARTMENT OF COMMERCE**  
**Ronald H. Brown, Secretary**

**TECHNOLOGY ADMINISTRATION**  
**Mary L. Good, Under Secretary for Technology**

**NATIONAL INSTITUTE OF STANDARDS  
AND TECHNOLOGY**  
**Arati Prabhakar, Director**







## Table of Contents

### Volume II - Appendices

A. User Instructions .....	A-1
A.1 Step-By-Step Instructions .....	A-1
A.2 Data Entry Rules and Field Attributes .....	A-10
A.3 Visual Manual .....	A-12
B. Installation and Configuration .....	B-1
B.1 Client Workstation .....	B-1
B.2 File Server .....	B-5
C. System Error Messages .....	C-1
C.1 User Interface Error Messages .....	C-1
C.2 Search Engine Error Messages .....	C-20
D. Listing of the Code .....	D-1
D.1 User Interface Code .....	D-3
D.2 Search Engine Code .....	D-317
E. Scout Comments from EAMATE Testing Session .....	E-1
F. Preliminary Employer Report Statistics .....	F-1
F.1 Einstats File .....	F-1
F.2 Indexstats File .....	F-7

**NOTE:** The table of contents for Volume II is repeated here for ease of use.  
See Volume I for a full table of contents for NISTIR 5394.







## A User Instructions

Appendix A contains user instructions for the EAMATE application. It is divided into three sections. Section A.1 contains step-by-step instructions for executing every feature of the application. Section A.2 lists data entry rules and field attributes, and section A.3 is a visual manual which provides a quick alternative to section A.1.

### A.1 Step-By-Step Instructions

Section A.1 lists instructions for each of the options available in the EAMATE application. Subheadings consist of the main screen (and credit screen) along with each of the five main screen choices. Detailed instructions for each feature are listed in the appropriate section along with pertinent notes and hints.

#### Main Application Screen and Credit Screen:

When the application is opened, the first screen that appears is the credit screen. To make this screen disappear, simply press the enter key or move the mouse pointer onto the Continue push-button and press the left mouse button once.

Now the application is ready to be used. The main screen of the application offers five push-button choices: Single Query, Report Statistics, Browse Report, Print Report and Exit.

To access any of the features on the main screen simply move the mouse pointer onto the desired push-button and press the left mouse button once.

The **Single Query** option provides the mechanism for searching for a particular individual.

The **Report Statistics** option is not implemented at this time but is included here to demonstrate functionality.

The **Browse Report** option allows access to information in the same order as reported by the employer.

The **Print Report** option provides printing capability for an employer report with less than 5000 employees. To print a larger report, submit a request to the system administrator.

The **Exit** option quits the application.



### The Single Query Option:

After selecting the Single Query option from the main menu, a dialog box appears titled "Please Enter Query Information" and the caret is positioned in the first data entry field. Type in the **year** (for the prototype this is only 1991) and the caret will automatically jump to the next data entry field.

Once the year has been entered, enter the **Employee Identification Number (EIN)**. The dash will automatically be inserted in the appropriate position and when the data entry field is complete the caret will automatically move to the last name data entry field.

The **last name** and **first name** fields are optional. However, if one is entered the other must be. If no first or last name is entered then a social security number must be entered (see below). Once a last name is entered, press the tab key to move to the first name data entry field. Once a first name is entered press the tab key to move to the social security number data entry field.

The **social security number** data entry field is also optional. However, if no first and last name are entered, then a social security number must be entered. When entering a social security number the dashes will automatically appear in the appropriate position. When the data entry is complete, the focus will move to the OK push-button.

To request the search be performed, press the enter key or move the mouse pointer onto the OK push-button and press the left mouse button once. If invalid data has been entered, message boxes will appear indicating the problem. To clear the message boxes, press the enter key or move the mouse pointer onto the OK push-button in the message box and press the left mouse button once.

If an incorrect year is entered a message box will appear identifying the problem. Press the enter key or move the mouse pointer onto the OK button and press the left mouse button once to clear the message box. Backspace over the incorrect year and type in the correct year. Press the enter key or position the mouse pointer onto the OK push-button and press the left mouse button once to begin the search process again.

If the EIN does not exist, a message box will appear. Clear the message box and re-enter the EIN. Press the enter key or position the mouse pointer onto the OK push-button and press the left mouse button once to begin the search process again.

If invalid combinations of the first name, last name, and social security fields occur then a message box will appear indicating the problem. Clear the message box, re-enter legal parameters and re-try the search.

After the search is complete, the Query Info and Possible Matches screen will appear. See the next section for instructions for maneuvering through this screen.



### Query Info and Possible Matches Screen:

The Query Info and Possible Matches Screen provides access to employer and employee information including browse, detail, and total data. Employer browse information is displayed in the top left corner of the screen and employee browse information is displayed in the list box on the bottom half of the screen. Detailed and totals data are available from within the list box or by accessing one of the push buttons.

To access employer detail information, move the mouse pointer onto the **Employer Detail** push-button and press the left mouse button once. Two choices are offered on the employer detail screen. Press the Continue button with the mouse pointer and the employer detail screen will disappear, or press the Print button with the mouse pointer and the employer detail data will be sent to the printer and a message box will appear indicating that printing is in progress. Press the Continue button with the mouse pointer on the message box to clear the message box and select other options from the Query Info and Possible Matches Screen as desired.

NOTE: If the Now Printing Message Box does not respond that means the system is still preparing the data for printing. Once complete, the system will release this box and it will disappear.

To access a **different employer report** associated with a multiple-report EIN, position the mouse pointer onto the **Change icon** (the blue arrow) and press the left mouse button once. Answer the message prompt by entering the desired 3-character report number. Press OK to make the change. (Notice the new employer report in the employer browse information section. The employer detail has also been updated as a result of this operation.)

The employee browse information displayed in the list box is returned in a ranked order pertaining to the query parameters. In other words, the names and/or social security numbers will be listed in the order that most closely matches the query parameters. The list box information may be scrolled through using the up and down arrow keys, the scroll bars or the page up and page down keys. To use the scroll bar, position the mouse pointer on the up or down arrows in the scroll bar and hold down the left mouse button or slowly click the left mouse button. To use the scroll button (the gray rectangle in the scroll bar), position the mouse pointer on the scroll button and while holding down the left mouse button, drag the mouse up and down as desired.

If the employee is not found in the first set of 50 matches returned in the list box, the next set of 50 possible matches may be accessed by pressing the **Additional Matches** button with the mouse pointer. The next set of fifty matches will then be added to the list box. The highlight bar will be positioned on the first entry of the new set of fifty to facilitate user browsing.

NOTE: Due to memory constraints and the fact that this is a prototype, the list box will hold approximately 700 employee records depending on the size of the record. Once the limit has been reached, no records will be added to the list box. If this situation occurs, a message box will appear indicating the occurrence.



To access the **employee detail data**, move the highlight bar in the list box with the up and down arrow keys to the desired selection and press enter, or position the highlight bar with the mouse pointer and double-click on the highlight bar to bring up the employee detail information.

Two choices are offered on the employee detail screen. Press the Continue button with the mouse pointer and the employee detail screen will disappear, or press the Print button with the mouse pointer and the system will prompt the user to enter the number of employee details to print. Enter the desired number and press the OK button with the mouse pointer. The employee detail data will be sent to the printer and a message box will appear indicating that printing is in progress. Press the Continue button with the mouse pointer on the message box to clear the message box and select other options from the Query Info and Possible Matches Screen as desired.

NOTE: If the Now Printing Message Box does not respond that means the system is still preparing the data for printing. Once complete, the system will release this box and it will disappear.

To access final totals information for an EIN, press the **Report Totals** push-button with the mouse pointer. Two choices are offered on the Totals screen. Press the Continue button with the mouse pointer and the Totals screen will disappear, or press the Print button with the mouse pointer and the report totals data will be sent to the printer and a message box will appear indicating that printing is in progress. Press the Continue button with the mouse pointer on the message box to clear the message box and select other options from the Query Info and Possible Matches Screen as desired.

NOTE: If the Now Printing Message Box does not respond that means the system is still preparing the data for printing. Once complete, the system will release this box and it will disappear.

Another option on the Query Info and Possible Matches screen is the Potential Blanket push-button and it involves identification of a possible blanket. If a possible blanket adjustment is identified, press the **Potential Blanket** push-button with the mouse pointer. A question screen will appear. Answer the questions per the on-screen instructions and proceed. Next, a screen will appear prompting you to enter the 3-character report number of the EIN. Enter the report number and press the OK button with the mouse pointer. The Query Info and Possible Matches screen will disappear and the Blanket Information screen will appear. To cancel this operation before going to the Blanket Information screen press the Cancel button with the mouse pointer and the screen that prompts for the EIN identifier will disappear.

If the report number can not be found, a message box will appear. To clear the message box press the OK button with the mouse pointer and re-enter the report number. Once the report number is entered, press the OK button with the mouse pointer to access the Blanket Information Screen.



To view the most recently entered **query parameters**, position the mouse pointer onto the **QP icon** and press the left mouse button once. A screen will appear containing the most recently entered query parameters. To clear the screen press the OK button with the mouse.

To exit the Query Info and Possible Matches Screen, press the **Close** push-button with the mouse pointer and the Query Information and Possible Matches Screen will disappear and the main application screen will reappear allowing for further selections.

Before the Query Info and Possible Matches Screen disappears, a question screen will appear prompting the user to enter information about the current query operation before moving on to other operations. Answer the questions by following the on-screen instructions.

#### Blanket Information Screen:

The Blanket Information Screen offers the same types of information as the Query Info and Possible Matches Screen except that it displays 30 records - ten from the beginning, ten from the middle, and ten from the end - of the specified employer report instead of possible matches.

To access the employer detail data, press the **Employer Detail** button with the mouse pointer and the employee detail screen will appear. Two choices are offered on the employer detail screen. Press the Continue button with the mouse pointer and the employer detail screen will disappear, or press Print with the mouse pointer and the employer detail data will be sent to the printer and a message box will appear indicating that printing is in progress. Press the Continue button with the mouse pointer on the message box to clear the message box and select other options from the Blanket Information Screen as desired.

NOTE: If the Now Printing Message Box does not respond that means the system is still preparing the data for printing. Once complete, the system will release this box and it will disappear.

To access final totals information for an EIN, press the **Report Totals** push-button with the mouse pointer and the Totals screen will appear. Two choices are offered on the Totals screen. Press the Continue button with the mouse pointer and the Totals screen will disappear, or press the Print button with the mouse pointer and the report totals data will be sent to the printer and a message box will appear indicating that printing is in progress. Press the Continue button with the mouse pointer on the message box to clear the message box and select other options from the Blanket Information Screen as desired.

NOTE: If the Now Printing Message Box does not respond that means the system is still preparing the data for printing. Once complete, the system will release this box and it will disappear.

To print a blanket report, press the **Print Blanket** push-button with the mouse pointer. A message box will appear indicating that printing is in progress. Press the Continue button with



the mouse pointer on the message box to clear the message box and select other options from the Blanket Information Screen as desired.

NOTE: If the Now Printing Message Box does not respond that means the system is still preparing the data for printing. Once complete, the system will release this box and it will disappear.

The list box containing employee browse information may be scrolled through using the up and down arrow keys, the scroll bars or the page up and page down keys. To use the scroll bar, position the mouse pointer on the up or down arrows in the scroll bar and hold down the left mouse button or slowly click the left mouse button. To use the scroll button (the gray rectangle in the scroll bar), position the mouse pointer on the scroll button and while holding down the left mouse button, drag the mouse up and down as desired.

To access the **employee detail data**, move the highlight bar in the list box with the up and down arrow keys to the desired selection and press enter, or position the highlight bar with the mouse pointer and double-click on the highlight bar to bring up the employee detail information.

Two choices are offered on the employee detail screen. Press the Continue button with the mouse pointer and the employee detail screen will disappear, or press the Print button with the mouse pointer and the system will prompt you to enter the number of employee details to print. Enter the desired number and press the OK button with the mouse pointer. The employee detail data will be sent to the printer and a message box will appear indicating that printing is in progress. Press the Continue button with the mouse pointer on the message box to clear the message box and select other options from the Blanket Information Screen as desired.

NOTE: If the Now Printing Message Box does not respond that means the system is still preparing the data for printing. Once complete, the system will release this box and it will disappear.

To exit the Blanket Information Screen, press the **Close** push-button with the mouse pointer and the Blanket Information Screen will disappear and the main application screen will reappear allowing for further selections.

#### The Report Statistics Option:

As stated above, this option is included to demonstrate functionality and is not implemented at this time. However, a query parameter screen and a place holder screen are included. If you select the Report Statistics Option, you will be prompted to enter the year, EIN, and establishment number. Press the OK button with the mouse pointer to bring up the Report Statistics Screen or press the Cancel button with the mouse pointer to return to the main application screen.



### Report Statistics Screen:

The Report Statistics Screen is included here as a place holder. This function is not implemented for the prototype. To clear this screen and return to the main application screen, press either the OK button or the Cancel button with the mouse pointer and the Report Statistics Screen will disappear.

### The Browse Report Option:

When the Browse Report Option is selected an entry screen will appear prompting you to "Please Enter the Browse Report Information." The required data entry fields for this screen are: the year, the EIN, and the beginning Microfilm Reference Number (MRN). The Establishment Number data entry field is optional.

When the prompt screen first appears, the caret will be located in the year data entry field. Type in the **year** and the caret will automatically move to the EIN data entry field.

Next type in the **EIN**. The dash will automatically appear at the correct position. The caret will automatically jump to the Beginning MRN data entry field.

Enter the **Beginning MRN** and the browse report query information is complete. To access the Browse Report Information Screen, press the OK button with the mouse pointer and the prompt screen will disappear and the Browse Report Information Screen will appear. To cancel the operation, press the Cancel button with the mouse pointer and the prompt screen will disappear, and the main application screen will reappear.

If the year is incorrect, a message box will appear indicating the problem. To clear the message box press the OK button with the mouse pointer and re-enter the year. Once the year is entered, press the OK button to access the Browse Report Information Screen.

Similarly, if the EIN can not be found, a message box will appear. To clear the message box press the OK button with the mouse pointer and re-enter the EIN. Once the EIN is entered, press the OK button to access the Browse Report Information Screen.

If the Beginning MRN does not exist, a message box will appear. To clear the message box press the OK button with the mouse pointer and re-enter the Beginning MRN. Once the EIN is entered, press the OK button to access the Browse Report Information Screen.

Follow the instructions in the next section for maneuvering through the Browse Report Information Screen.



### Browse Report Information Screen:

The Browse Report Information Screen is organized in a similar fashion to the Query Info and Possible Matches Screen and the Blanket Information Screen. Employer browse information is located in the top left corner and employee browse information is located in the list box on the bottom half of the screen. Option push buttons are located down the right side of the screen.

To access the employer detail data, press the **Employer Detail** button with the mouse pointer and the employer detail screen will appear. Two choices are offered on the employer detail screen. Press the Continue button with the mouse pointer and the employer detail screen will disappear, or press Print with the mouse pointer and the employer detail data will be sent to the printer and a message box will appear indicating that printing is in progress. Press the Continue button with the mouse pointer on the message box to clear the message box and select other options from the Blanket Information Screen as desired.

NOTE: If the Now Printing Message Box does not respond that means the system is still preparing the data for printing. Once complete, the system will release this box and it will disappear.

To access final totals information, press the **Report Totals** push-button with the mouse pointer and the Totals screen will appear. Two choices are offered on the Totals screen. Press the Continue button with the mouse pointer and the Totals screen will disappear, or press the Print button with the mouse pointer and the report totals data will be sent to the printer and a message box will appear indicating that printing is in progress. Press the Continue button with the mouse pointer on the message box to clear the message box and select other options from the Browse Report Information Screen as desired.

NOTE: If the Now Printing Message Box does not respond that means the system is still preparing the data for printing. Once complete, the system will release this box and it will disappear.

The list box containing employee browse information may be scrolled through using the up and down arrow keys, the scroll bars or the page up and page down keys. To use the scroll bar, position the mouse pointer on the up or down arrows in the scroll bar and hold down the left mouse button or slowly click the left mouse button. To use the scroll button (the gray rectangle in the scroll bar), position the mouse pointer on the scroll button and while holding down the left mouse button, drag the mouse up and down as desired.

Fifty employee records are added to the list upon initial opening of the Browse Report Information Screen. To add records to the list box, press the **Additional Records** push-button with the mouse pointer and fifty new records will be added to the list box. The highlight bar will be positioned on the first entry of the new set of fifty to facilitate user browsing.

NOTE: Due to memory constraints and the fact that this is a prototype, the number of records that can be added to the list box is approximately 700.



**HINT:** If you exceed 700 records and want to look farther in the report, close out the current browse, and begin a new browse. Enter a higher MRN so that the starting point for the browse is farther along in the report.

To access the **employee detail data**, move the highlight bar in the list box with the up and down arrow keys to the desired selection and press enter, or position the highlight bar with the mouse pointer and double-click on the highlight bar to bring up the employee detail information.

Two choices are offered on the employee detail screen. Press the Continue button with the mouse pointer and the employee detail screen will disappear, or press the Print button with the mouse pointer and the system will prompt you to enter the number of employee details to print. Enter the desired number and press the OK button with the mouse pointer. The employee detail data will be sent to the printer and a message box will appear indicating that printing is in progress. Press the Continue button with the mouse pointer on the message box to clear the message box and select other options from the Browse Report Information Screen as desired.

**NOTE:** If the Now Printing Message Box does not respond that means the system is still preparing the data for printing. Once complete, the system will release this box and it will disappear.

To exit the Browse Report Information Screen, press the **Close** push-button with the mouse pointer and the Browse Report Information Screen will disappear and the main application screen will reappear allowing for further selections.

#### The Print Report Option:

Once the print report option has been selected, the Print Report Information screen will appear. Password, year, EIN, and the three-character report number are required. The Establishment Number is optional.

When the screen appears, the caret will be located in the **password** data entry field. Type in the password and the caret will automatically jump to the year data entry field.

Next, type in the **year** and the caret will automatically jump to the EIN data entry field. Type in the **EIN** and the caret will automatically jump to the report number field. The dash in the EIN will automatically appear in the appropriate position.

Type in the **report number** and then press the OK button with the mouse pointer to send the specified report to the server printer. To cancel the operation before sending the information to the server, press the Cancel button with the mouse pointer.

**NOTE:** Only reports containing less than 5000 records may be printed. To print larger reports, submit a request to the system administrator.



If the year is incorrect, a message box will appear alerting the user. To clear the message box press the OK button with the mouse pointer and re-enter the year. Once the year is entered, press the OK button with the mouse pointer to send the print information to the server.

Similarly, if the EIN can not be found, a message box will appear alerting the user. To clear the message box press the OK button with the mouse pointer and re-enter the EIN. Once the EIN is entered, press the OK button with the mouse pointer to send the print information to the server.

If the report number can not be found, a message box will appear alerting the user. To clear the message box press the OK button with the mouse pointer and re-enter the report number. Once the report number is entered, press the OK button with the mouse pointer to send the information to the server.

Upon successful communication with the server, a message box will appear indicating that printing is in progress. Press the Continue button with the mouse pointer on the message box to clear the message box and select other options from the main application screen as desired.

NOTE: If the Now Printing Message Box does not respond that means the system is still preparing the data for printing. Once complete, the system will release this box and it will disappear.

#### The Exit Option:

After selecting the exit option from the main application screen, a message box will appear with the message "Are You Sure You Want To Exit?" Press the Yes button with the mouse pointer to quit the application, or press the No button with the mouse pointer to keep the application open and make the next desired selection.

## **A.2 Data Entry Rules and Field Attributes**

Section A.2 outlines data entry rules and describes field attributes for the items contained in the data entry screens. The table below lists the name of each data entry field, the field type (all fields are character-based, type of field refers to which characters are legal - either alpha, numeric, or both), maximum allowable length of the field in characters (maximum number of



characters that may be entered), the minimum required length of the field in characters (the minimum amount of characters required for the field to be valid), and whether the field is required. Special notes for required combinations of data entry fields along with other unique field-related information follow the table.

<u>Field Name</u>	<u>Field Type</u>	<u>Maximum Allowable Length</u>	<u>Minimum Required Length</u>	<u>Required or Optional</u>
<i>Single Query Screen</i>				
Year	numeric	4	4	required*
EIN	numeric	10	10	required
Estab. No.	numeric	4	4	optional
Last Name	alpha	15**	1	optional***
First Name	alpha	12**	1	optional***
SSN	numeric	11	11	optional***
<i>Report Statistics Screen</i>				
Year	numeric	4	4	required
EIN	numeric	10	10	required
Estab. No.	numeric	4	4	optional
<i>Browse Report Screen</i>				
Year	numeric	4	4	required
EIN	numeric	10	10	required
Estab. No.	numeric	4	4	optional
Beg. MRN	numeric	11	11	required
<i>Print Report Screen</i>				
Password	alphanumeric	5	5	required
Year	numeric	4	4	required
EIN	numeric	10	10	required
Estab. No.	numeric	4	4	optional
Report No.	alpha	3	3	required
<i>Miscellaneous Screens</i>				
Num. Copies	numeric	5	1	optional****
Report No.	numeric	3	3	optional*****

Table A-1 Data Entry Fields and Their Attributes



### Special Notes

- \* For the prototype, the year field only accepts the year 1991. Other values will cause an error box to appear.
- \*\* You may enter up to 27 characters in both the first name and last name fields. However, only 12 characters will be recognized in the first name field and only 15 characters will be recognized in the last name field. Excess characters will be truncated.
- \*\*\* While the first name field, the last name field, and the SSN field are all listed as optional, obviously some or all must be entered so a search may be initiated. The required legal combinations are:
  - first name and last name and SSN
  - first name and last name
  - SSN

One of these combinations MUST BE entered in order to request a search on an individual. If one of the above legal combinations is not entered on the single query data entry screen before initiating a search, error and/or message boxes will appear.

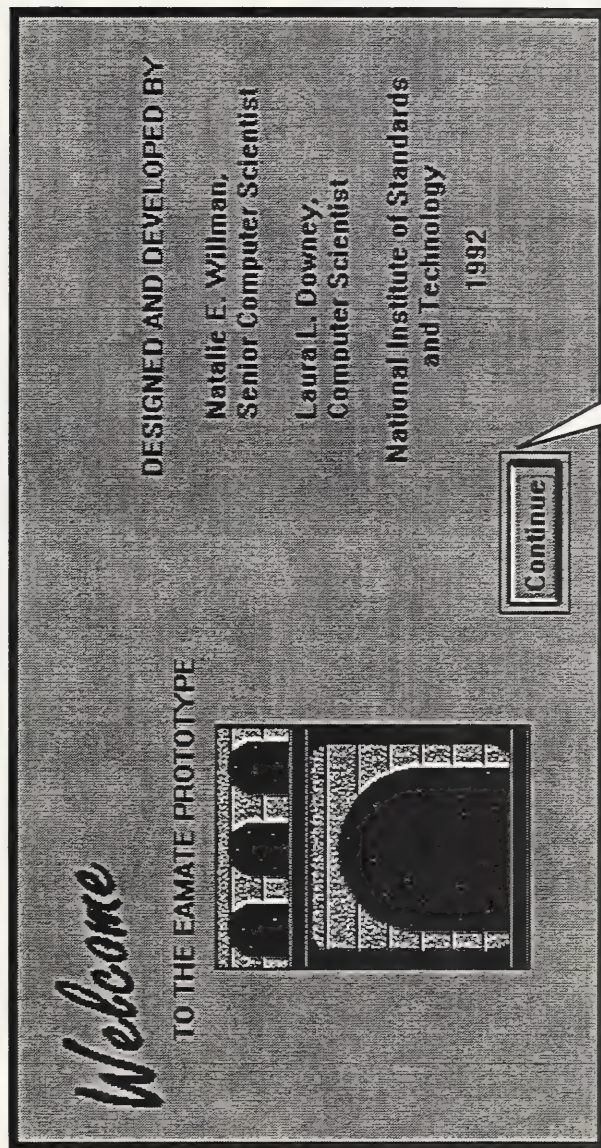
- \*\*\*\* The number of copies field will appear when a detail print request is made. The field is listed as optional because if nothing is entered the number of copies to print will default to one.
- \*\*\*\*\* When changing the employer browse information (choosing the Change icon), or when making a potential blanket selection, a message box will appear requesting entry of the 3-character report number. If no report number is entered, the current report number will be used as the default.

## **A.3 Visual Manual**

Section A.3 depicts the major screens incorporated into the EAMATE application in an annotated visual format. This section is intended as an alternative to traditional step-by-step instructions of the type found in section A.1. The visual manual offers a quick and easy overview of primary operations and their functions utilized in the EAMATE application and provides a mechanism for learning to use the application in the shortest time possible.



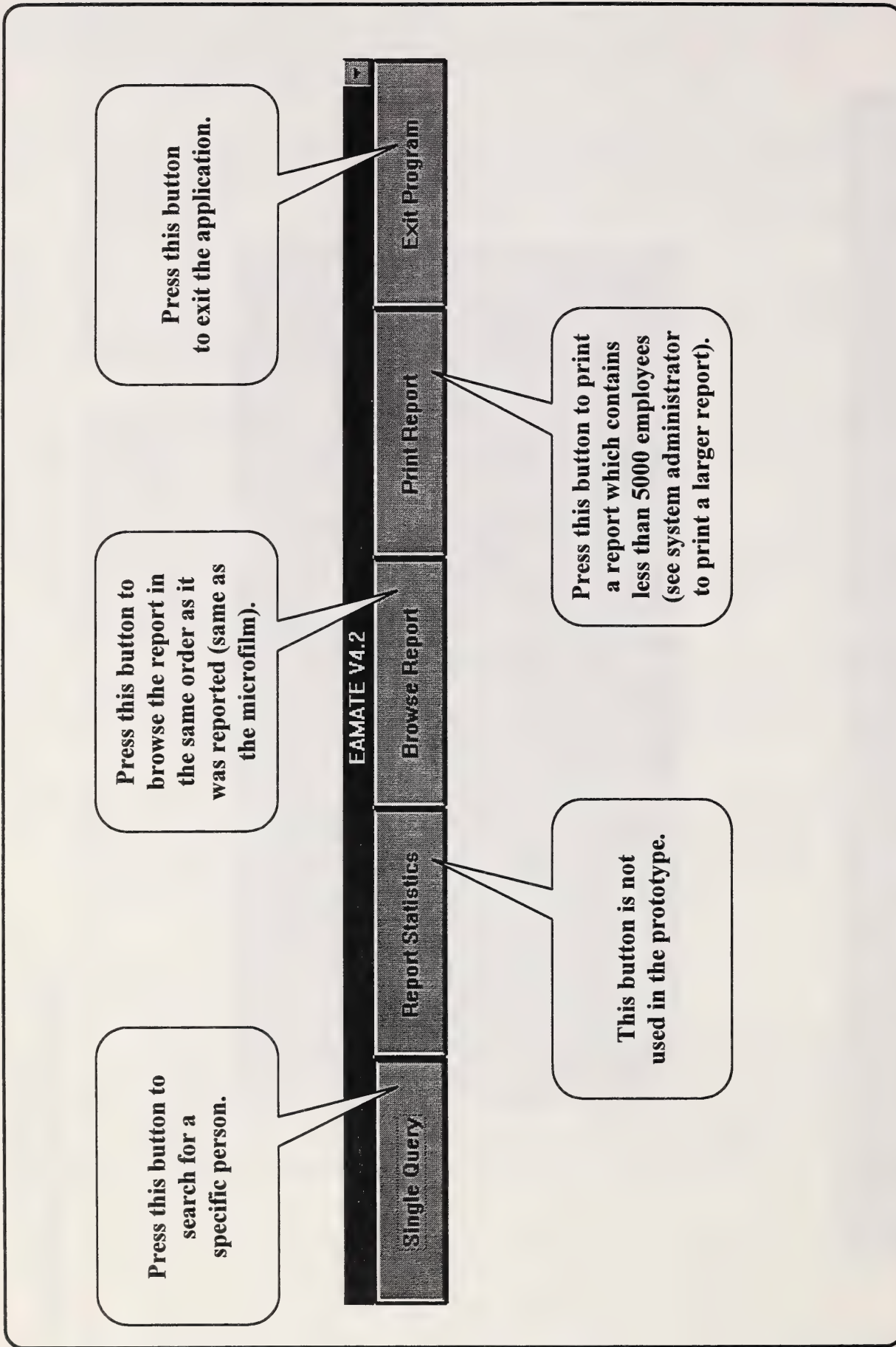
Single Query	Report Statistics	Browse Report	Print Report	Exit Program
--------------	-------------------	---------------	--------------	--------------



Press this button  
to remove the  
Welcome Screen.

The Opening Screen





The Main Screen



**Please Enter Query Information**

YEAR					
EIN					
ESTAB#					
LAST NAME					
FIRST NAME					
SSN					

**OK** **Cancel**

**You must enter the year 1991.**

**You must enter the EIN.**

**This field is optional.**

**Press OK after you finish entering the query parameters to begin the search.**

**Press this button to cancel entering the query parameters and to return to the main screen for another selection.**

**You must enter a combination of these three fields before beginning a search.**  
**Legal combinations are:**  
**First Name & Last Name**  
**OR**  
**First Name, Last Name & SSN**  
**OR**  
**just the SSN.**

The Single Query Prompt Screen

**Enter Browse/Report Information**

YEAR	EIN	ESTAB#	BEGINNING MRN
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

**Press OK after you finish entering the browse report parameters to bring up the browse information screen.**

**You must enter the year 1991.**

**You must enter the EIN.**

**This field is optional.**

**You must enter the beginning MRN.**

**Press this button to cancel entering the browse report parameters and to return to the main screen for another selection.**

The Browse Report Prompt Screen



Please Enter Print Information

PASSWORD

YEAR

EIN

ESTAB#

REPORT#

OK

Cancel

You must enter the year 1991.

You must enter the password.

Press OK after you finish entering the print report parameters to send the print request to the server.

Press this button to cancel entering the print report parameters and to return to the main screen for another selection.

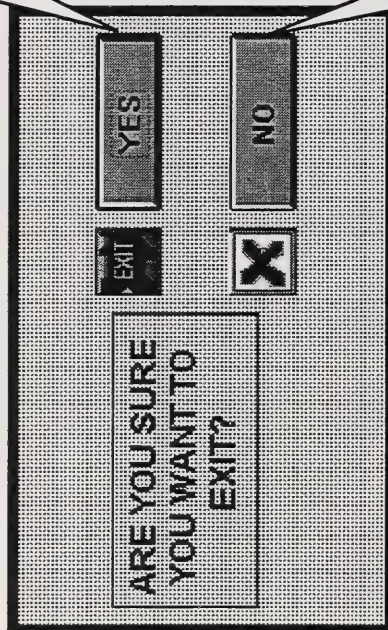
You must enter the beginning MRN.

You must enter the EIN.

This field is optional.

The Print Report Prompt Screen

Press this button if you want to exit the application.



Press this button if you do not want to exit the application. The Exit Screen will disappear and you may make another selection from the main screen.

The Exit Program Prompt Screen







## The Potential Blanket Information Screen







Press the OK button when you are finished with this screen.

## Please Answer the Following Questions

Did You Find the Person You Were Looking For?

☒ Yes  
☐ No

Were You Interrupted at Any Time During Your Search?

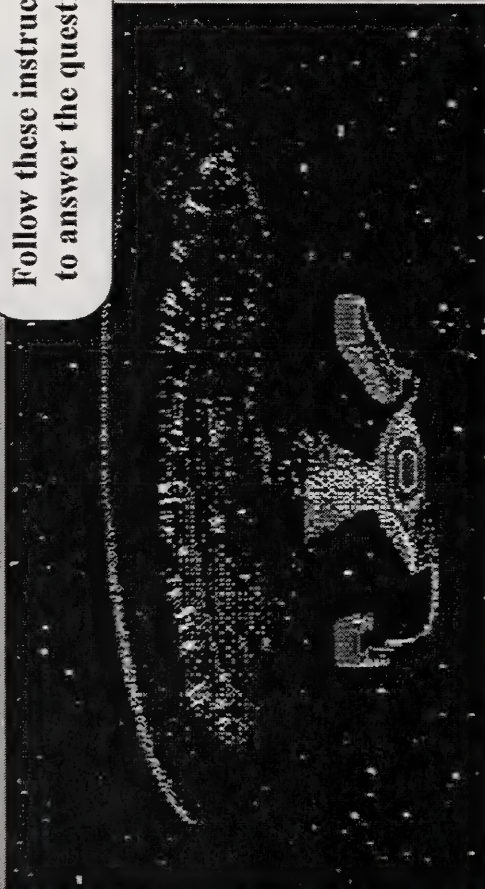
☐ Yes  
☒ No

Choose Yes or No to Each Question by Moving the Mouse Pointer onto the Circle Located Next to Desired Choice and Pressing the Left Mouse Button Once

OK

Press OK When You Are Finished Answering the Questions

Follow these instructions to answer the questions.



Beam Me Up Scotty!!! \*\*\*\*\* I'm Through Collecting Data

The Case Question Screen



## **B Installation and Configuration**

The following sections describe the setup and configuration necessary on the EAMATE prototype. The information in this section should be supplemented by the reference manuals supplied by the vendors of the installed products, and by other sections of this document, as specified in the instructions.

### **B.1 Client Workstation**

Section B.1 provides instructions for software installation and configuration of the EAMATE V4.2 application onto the client workstations that are part of the EAMATE Expanded Prototype (EEP). Each 486/66 workstation will already contain the latest versions of DOS, Microsoft Windows, PC-NFS, and will be connected to a LAN.

Each client workstation will require minor configuration changes to Microsoft Windows and PC-NFS in addition to the installation and configuration of the application, including associated work files, and a set of utilities for working with statistics gathered during application use. A 3-1/2" disk marked "EEP EAMATE V4.2 Install Files" is included as part of this appendix.

Additionally, this section provides instructions for initializing the environment for statistics-gathering operations as well as descriptions for accessing the recorded statistics.

### **Getting Started**

To begin, make sure you are at the root directory (the top level) in DOS.

Next, make a directory titled **EAMATE**.

Change to the **C:\EAMATE** directory.

Insert the disk titled "**EEP EAMATE V4.2 Install Files**" into the 3-1/2" drive.

Now you are ready to begin the installation.

### **Installation**

Copy the file **eamat42.exe** from the disk to the directory **C:\EAMATE**.

Next, copy the file **mfedit.ini** from the disk to the directory **C:\EAMATE**.

Next, copy the file **init.exe** from the disk to the directory **C:\EAMATE**.

Next, copy the file **astat.exe** from the disk to the directory **C:\EAMATE**.

Next, copy the file **whatuser.exe** from the disk to the directory **C:\EAMATE**.

At this point you are ready to copy **usenum.fil** which contains the user number assigned to the client. The user number is used by the server to determine "who" is making a request. **IT IS VERY IMPORTANT THAT EACH CLIENT WORKSTATION HAS A UNIQUE USER NUMBER OR THE APPLICATION WILL NOT EXECUTE PROPERLY.** The installation disk contains several copies of **usenum.fil** and each copy of **usenum.fil** contains a different user number. To install a specific **usenum.fil**, copy **usenum.fil** from the appropriate directory located on the installation disk. **USER1** directory contains **usenum.fil** with user number 1, **USER2** directory contains **usenum.fil** with user number 2, and so on.

For example, to copy **usenum.fil** containing user number 5, type:

copy **#:\USER5\usenum.fil** where # is the designated drive

NOTE: Make sure you are in the **C:\EAMATE** directory.

Now that you have installed the application file, the associated work files and the utility files, you are ready to install one required system file (a dynamic link library).

Change to the directory **C:\WINDOWS\SYSTEM**.

Next, copy the file **mfedit.dll** from the disk to the directory **C:\WINDOWS\SYSTEM**.

Installation is now complete and you are ready to make some configuration changes.

## **Configuration**

The EAMATE V4.2 application makes use of color as one of its features to distinguish data and location within the application. Some of the application settings are achieved by choosing a specific color scheme within Microsoft Windows. To set the appropriate color scheme, follow the steps listed below:

Change to the directory **C:\WINDOWS**.

Copy **color.ini** from the disk to **C:\WINDOWS**.

Change to the root directory **C:\**.

Open Microsoft Windows by typing **win** at the DOS prompt.

Open File Manager and locate the file **control.ini**.



Open the file **control.ini** and delete the sections title **[current]**, **[color schemes]**, and **[custom colors]**.

Save and exit the file **control.ini**.

Open the file **color.ini** and copy the contents into the clipboard and then close the file **color.ini**.

Open the file **control.ini** and position the cursor at the very beginning then paste the contents of the clipboard into the file **control.ini**.

Save and exit the file **control.ini**.

Exit the File Manager, the appropriate color scheme is now set.

Now you are ready to configure the **EAMATE** program group.

From the **File Menu** in **Program Manager**, select **New**.

A dialog box will appear prompting you to identify the current operation as a program group or a program object.

Choose the **program group** and press OK. Fill in the description box with **EAMATE**, leave the group file box blank, and press OK again. The new program group titled **EAMATE** will appear.

From the **File Menu** in **Program Manager**, select **New**.

Choose **program object** and press OK. A dialog box will appear prompting you to fill in pertinent information about the new program object you are creating. Fill in the information as described below:

Description:	<b>EAMATE SVGA V4.2</b>
Command Line:	<b>C:\EAMATE\eamat42.exe</b>
Working Directory:	<b>C:\EAMATE</b>

Press the OK button and size the **EAMATE** program group and place it in a prominent position on the desktop so that it will be easy to locate by the users. Make sure you save the current settings before exiting Microsoft Windows.

Exit Microsoft Windows and make sure you are at the root directory.

The final configuration change involves PC-NFS. The directory **/export/home/ssa** located on the server contains files that are accessed by both the user interface and the search engine. In order

for the user interface to use this directory, it must be mounted as a drive on the PC using PC-NFS.

Change to the directory **C:\NFS**.

Type **nfscnf** at the DOS prompt and the NFS Configuration program will appear.

Choose **Define Drive** under the **Resources Menu**.

Fill in the prompts as listed below:

Server:       **demeter**

Path:         **/export/home/ssa**

Sharing:      **ns**

Answer **NO** to the question, "Do you want to change the data transfer options?"

Hit return.

Press **Y** in response to the question about mounting the resource.

Next, you will be prompted to specify a drive letter - choose **d**:

Hit enter twice to mount the resource and return to the NFS configuration menu.

Exit the configuration menu.

**NOTE:**    Make sure that the server is set to the SPARCcenter 1000 where the data is stored.  
              The name of the server should be demeter.

The installation and configuration of the **EAMATE V4.2** application onto the client workstations that are part of the **EEP** is now complete.

### **Initializing the Statistics Environment**

After the application is installed, setup, and configured, the statistical environment needs to be initialized. To do this, perform the following steps:

Make sure you are in the DOS environment, not Microsoft Windows.



Change to the **C:\EAMATE** directory.

Type **init** at the DOS prompt.

The file **stat#.fil** has now been initialized (to all zeroes) and will be used when the application is opened to initialize the application for statistics-gathering operations. The program **init** only needs to be run one time during the installation process. Once initialized, each subsequent opening of the application will use the last set of statistics as a starting point thereby providing a running total (aggregate) of the statistics that may be accessed at any given time. If **init** is run again, the statistics which have been gathered to this point will be deleted.

### **Accessing the Statistics**

The statistics are recorded in the binary file **stat#.fil**. To access the statistics in ASCII format, perform the following steps:

Make sure you are in the DOS environment, not Microsoft Windows.

Change to the **C:\EAMATE** directory.

Type **astat** at the DOS prompt and follow the program prompts, entering the appropriate file name. The result of the program will be a file named **stat#.txt** and it will contain the aggregate statistics in an ASCII format.

For example, if the workstation is assigned user number 4, then you want to access the binary file **stat4.fil** to create the text file **stat4.txt**.

One other miscellaneous utility program exists which may be used to determine the user number. To execute this program, perform the following steps:

Make sure you are in the DOS environment, not Microsoft Windows.

Change to the **C:\EAMATE** directory.

Type **whatuser** at the DOS prompt.

The user number will be displayed on the screen.

## **B.2 File Server**

This section outlines the steps that are necessary in order to configure the file server for use with the EAMATE applications. The steps in this section should be supplemented with the operating

system and system administrator reference guides available with Solaris 2.x, the on-line "man" pages on the file server system, and sections 6.3, 7, and 8 of this document.

NOTE: The file server host name should be set to "demeter", as this is the file server name which is hard coded into the RPC calls at several locations.

NOTE: The user account should be created as the super user ("root"), and the remainder of the steps should be taken while logged into the "ssa" account.

## **1. Create A User Account**

*References: Solaris 2.x documentation, "man" page for "admintool" and "passwd"*

The file server configuration relies on having an installed user account where all of the data is stored, and from which all of the data conversion, indexing and searching applications are run.

This account is created, while logged in as "root", by using the "User Account Manager" section of the "admintool" UNIX utility. The user name should be "ssa", using the next consecutive user id over 100. The login shell should be the C shell (/bin/csh), and the home directory should be /export/home/ssa. The account should be password protected, which can be accomplished by use of the "passwd" command.

## **2. Create Directory Hierarchy**

*References: Solaris 2.x documentation, "man" page for "share" and "vfstab", section 7.x of this document*

Once the account is created, it is necessary to create the directory hierarchy necessary to convert, index and search the data, and to maintain the source code. This directory hierarchy exists under the home directory of the "ssa" account. It is also vital that the home directory of the "ssa" account be exported for use by remote workstations. This can be accomplished by use of the "share" command and the use of the "/etc/dfs/dfstab" file. The following directories should be created:

Root Account Directory:	/export/home/ssa (DOES NOT NEED TO BE CREATED)
Root Project Directory:	/export/home/ssa/ssacode
Unparsed Data Directory:	/export/home/ssa/datahold
Root Browse Data Directory:	/export/home/ssa/1991BRW
Root Detail Data Directory:	/export/home/ssa/1991DET

Browse Data Mount Points:	/export/home/ssa/1991BRW/1 ...
	/export/home/ssa/1991BRW/n

NOTE: One sub directory should be created for each magnetic hard drive which will be mounted to the system to store the browse data. The system should be configured to mount these magnetic hard drives to these directories at boot time (using the



/etc/vfstab file). When the system is set up and mount points are chosen, an entire 2.1 GB drive should be mounted to the /export/home/ssa directory in order to have enough space to store the index and duplicate posting files, and the other 2.1 GB drives should be mounted to the browse data mount points. If all of the index and duplicate postings files cannot be stored on the single 2.1 GB disk mounted to the /export/home/ssa directory, they can be stored somewhere else. However, the files **MUST** be symbolically linked back to the /export/home/ssa directory in order for the searching applications to work.

Detail Data Mount Points:       /export/home/ssa/1991DET/1 ...  
                                  /export/home/ssa/1991DET/n

NOTE:     One sub directory should be created for each side of each platter which will be mounted to the system to store the detail data. The system should be configured to mount these platters to these directories at boot time (using the /etc/vfstab file).

### 3. Install the Source Code and Executables

*References: Solaris 2.x documentation, "man" page for "tar", sections 6.3 and 7.1 of this document*

Once the directory hierarchy is set up, then the source code can be installed from the 4mm DAT tape. The source code was backed up using the "tar" command, and a blocking factor of 1024. The data should be restored while in the "ssacode" sub directory. The command to restore the data is:

```
tar -xvfb <devicename> 1024
```

where <devicename> is the device name of the 4mm DAT tape drive (for example: /dev/rmt/1).

When this tape is restored in the /export/home/ssa/ssacode directory, the following sub directories will be created, and will contain the listed source code files. In addition, all relevant "Makefiles" will be copied to the directories listed below.

Include Files:                   ssacode/include  
                                  btreetstruct.h  
                                  eamateststruct.h  
                                  params.h  
                                  paramsemplr.h

Executable Files:               ssacode/bin  
                                  client  
                                  debug  
                                  download  
                                  fix\_parse  
                                  index  
                                  indexemplr

- parse
- search\_addmatch
- search\_blanket
- search\_browse
- search\_detail
- search\_header
- search\_print
- search\_single
- sysadm\_print

Library Files:

- ssacode/lib
- libbtree\_data.a
- libbtree\_emplr.a
- libgen\_eamate.a
- libtest.a

Source Files:

Library Source Files:

- ssacode/src
- ssacode/src/lib
- ./btree\_data/btree\_data.c
- ./btree\_emplr/btree\_emplr.c
- ./general/eamate.c
- ./general/general.c
- ./test/test\_funcs.c

Application Source Files:

- ssacode/src/bin
- ./client/client.c
- ./debug/debug.c
- ./download/download.c
- ./fix\_parse/fix\_parse.c
- ./index/index.c
- ./indexemplr/indexemplr.c
- ./parse/parse.c
- ./search\_addmatch/search\_addmatch.c
- ./search\_blanket/search\_blanket.c
- ./search\_browse/search\_browse.c
- ./search\_detail/search\_detail.c
- ./search\_header/search\_header.c
- ./search\_print/search\_print.c
- ./search\_single/search\_single.c
- ./sysadm\_print/sysadm\_print.c

Once the source code is copied into the directories, then it can be compiled and installed into the final directories. The "Makefiles" included with the source code will work for the SunPro compiler for Solaris 2.x. If a different compiler is used, some of the flags may need to be changed. The "Makefiles" have three options: "make bare", "make depend", and "make install".



All of these options can be run from any level of the hierarchy, but should be finally run from the root directory of the project hierarchy (/export/home/ssa/ssacode). The options are as follows:

- make bare  
This option is used to clean out old copies of executables and libraries before a new copy is made. The use of this option will clean out the ssacode/bin and the ssacode/lib directories, and will remove the executable and object code files from each of the source code sub directories. This will ensure that the next time "make install" is run, all of the source code will be recompiled.
- make depend  
This option is used to generate the lists of source code modules, libraries and include files which a given source code module depends upon. This will ensure that an executable will be recompiled every time that a module that it depends upon is modified.
- make install  
This option is used to actually compile any source code modules which are out of date with regard to the current executable. In addition, it will place the compiled libraries in the ssacode/lib directory, and the compiled executables in the ssacode/bin directory.

To install the source code, from the "ssacode" directory, a "make bare" should be issued, then a "make depend", and then a "make install". Once these steps are successfully executed, then the executable files in the ssacode/bin directory can be copied into the home directory of the user account (/export/home/ssa).

#### **4. Convert and Index the Data**

*References: Solaris 2.x documentation, section 7 of this document*

Once the directory hierarchy is set up, and the applications are installed, the data can be downloaded and converted. Basically, this entails transferring the unprocessed COM files to the /export/home/ssa/datahold directory, parsing the files into browse and detail data, and indexing the data. Step by step directions on how to complete this process can be found in section 7.2 of this document.

#### **5. Start the Search Applications**

*References: Solaris 2.x documentation, section 8 of this document*

Once the data is indexed, then the system is ready to be set up for requests from a remote workstation. This consists of starting the search applications, and step-by-step instructions for this process can be found in section 8 of this document.





## **C System Error Messages**

This Appendix details error messages that may be printed by the user interface code or the applications resident on the file server (data conversion, indexing and searching). Each error message is followed by an explanation of the error, and possible conditions which may generate the error.

### **C.1 User Interface Error Messages**

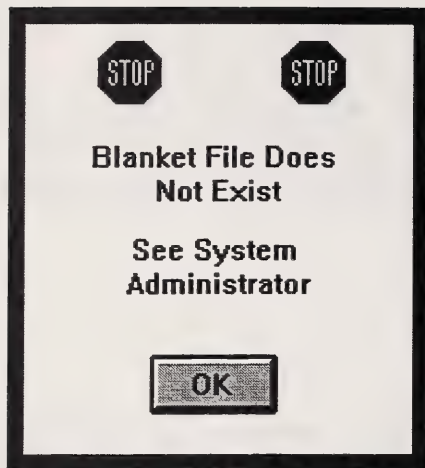
The EAMATE User Interface Prototype provides a variety of error trapping and reporting techniques and capabilities. These range from interactive message boxes and message beeps to specific messages recorded in a local file that resides on each client. Message beeps occur when the user attempts to input disallowed characters into an input field. For example, the SSN field will only accept digits, so if a user types a non-digit character the system will beep and the illegal character will not be recorded. The system will then wait for a legal character to be entered.

Message boxes appear when legal but invalid data has been entered and also when system errors occur. For example, if a user enters a legal first name but does not enter a last name, a message box will appear indicating that if a first name is entered, then a last name must also be entered. Message boxes also appear when system errors occurs such as an unlocatable file.

The third mechanism for indicating errors is the local error file (err.fil) located on each client. When particular errors occur, such as null file handles or communication errors, these problems, along with identifying information, are recorded in the local error file. When a problems occurs, the system administrator can merely look at the last message in the error file for help in trouble shooting. Below is a list of possible errors and corresponding error handling procedures and indicators with the exception of message beeps that deal with entry of illegal characters.

#### **C.1.1 Error Message Boxes**

This section displays all the error/message boxes incorporated into the user interface prototype and provides the name of the box (as contained in the dialog.dlg file) and the reason for its appearance.



BLANKERR appears when the application is unable to open the data file needed to create the blanket information screen.



BLANKETERR appears when the creation of a client handle fails during a request for blanket information or when the system fails to set the retry time out during the request for blanket information or if the RPC call fails during a request for blanket information.

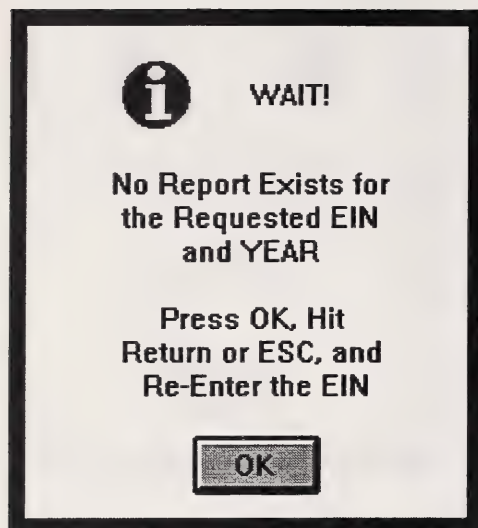




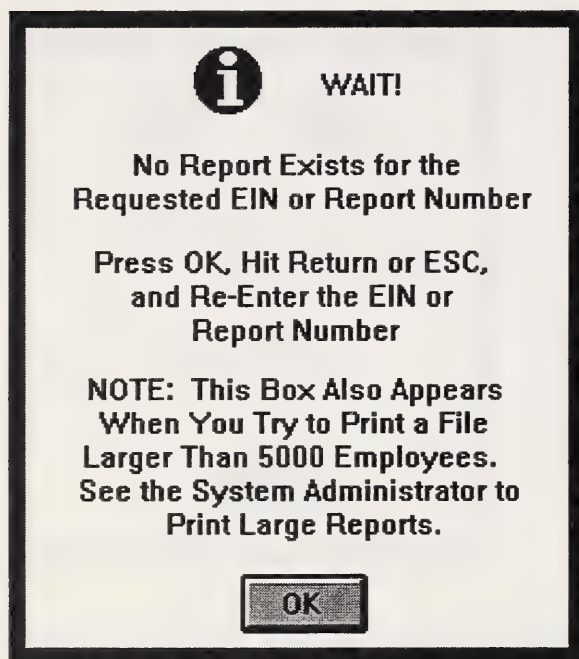
DATAERR appears when the application is unable to open the file containing browse information.



DETAILTXT appears when the application is unable to open the file containing detail information.



EINERROR appears when an incorrect EIN has been entered.



EINORSEQ appears when an incorrect EIN or an incorrect report number has been entered during a print request. It also appears if the file that has been requested for print contains more than 5000 employees.

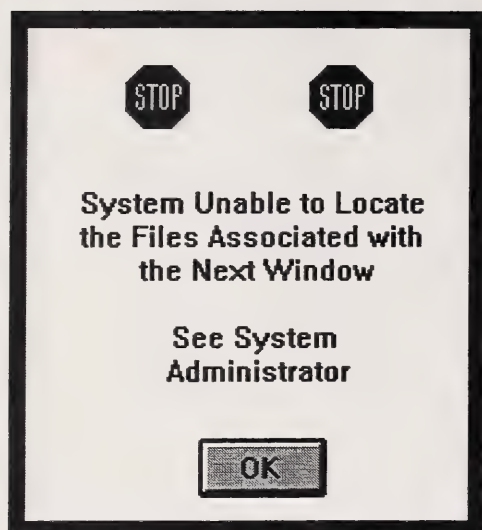




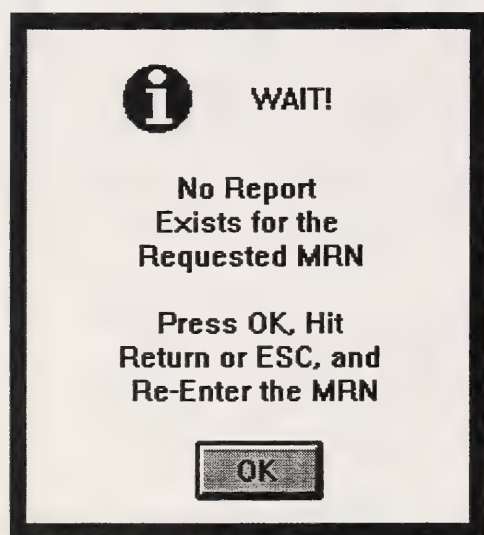
ERRORFILE appears when the application is unable to open the local error file.



HEADERTXT appears when the application is unable to open the employer information file.

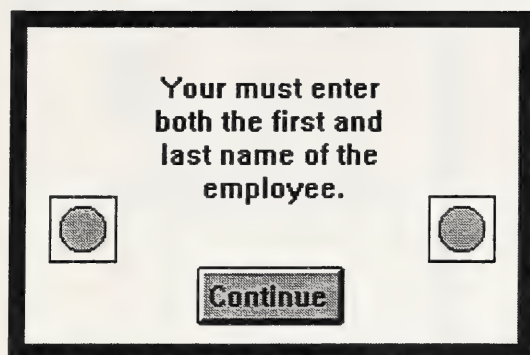


MISSINGFILE appears when the application is unable to open the employer information file and the browse information file before creating the query information screen or the browse information screen.



MRNERR appears when an incorrect MRN has been entered during a browse query.

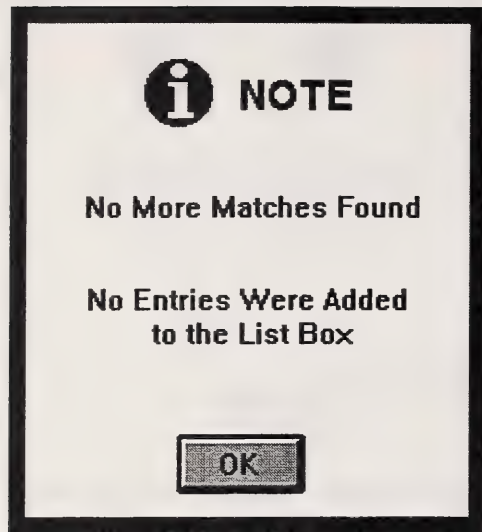




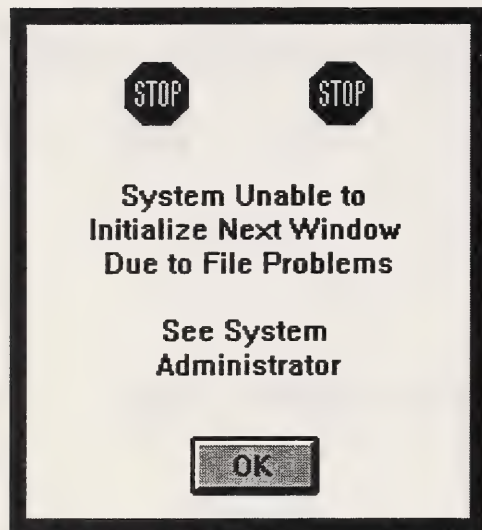
NMSG appears when a first name but no last name has been entered or vice versa.



NOMATCH appears when no approximate matches are returned for a target individual.



NOMORE appears when all additional matches have been displayed and no more remain.

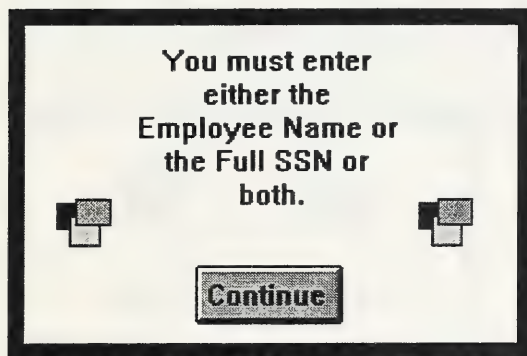


NULLPTR appears when the application is unable to open the employer information file and the blanket information file during initialization of the blanket information screen. It also appears when the application is unable to open the employer information file and the browse information file during the initialization of the query information screen.





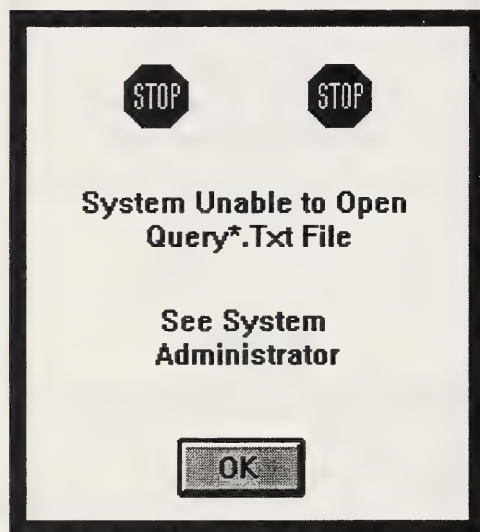
PWERROR appears when an incorrect password has been entered.



QMSG appears when an illegal combination of the name and social security fields has been entered.

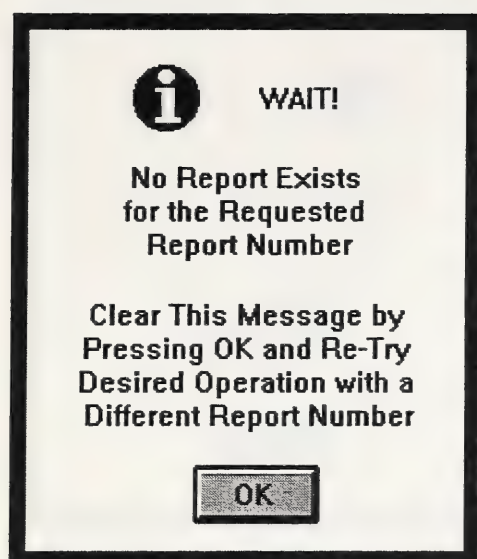


QUERYERROR appears as a result of a variety of situations. It is the basic generic system error message and indicates problems with remote procedure calls. Details of specific errors may be found in the local error file which should be accessed anytime QUERYERROR appears.

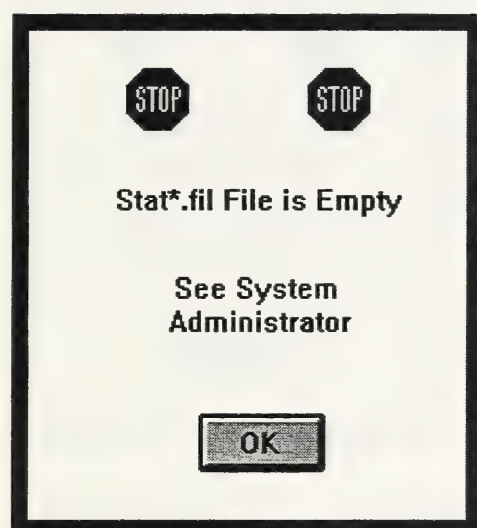


QUERYTXT appears when the application is unable to open the query information file.

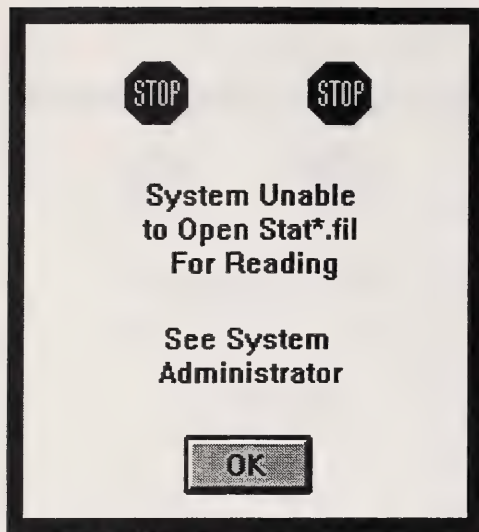




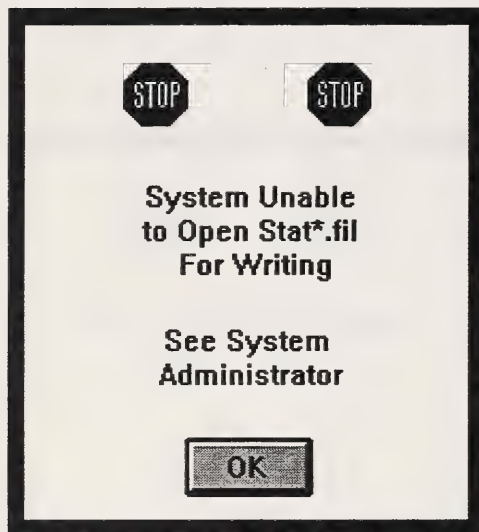
SEQERR appears when an incorrect report number has been entered.



STATEMPTY appears when the statistics file contains no data.

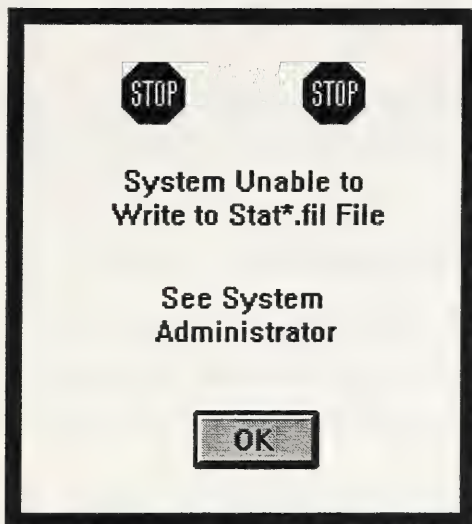


STATOPEN appears when the application is unable to open the statistics file in preparation for a read operation.

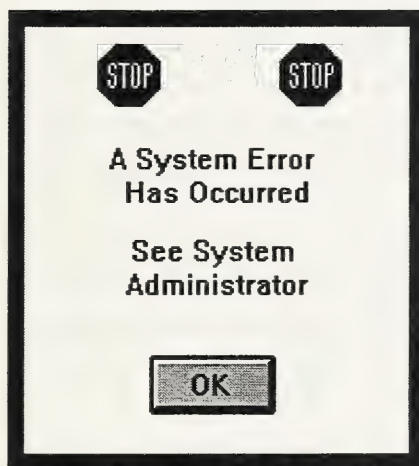


STATWOPEN appears when the application is unable to open the statistics file in preparation for a write operation.

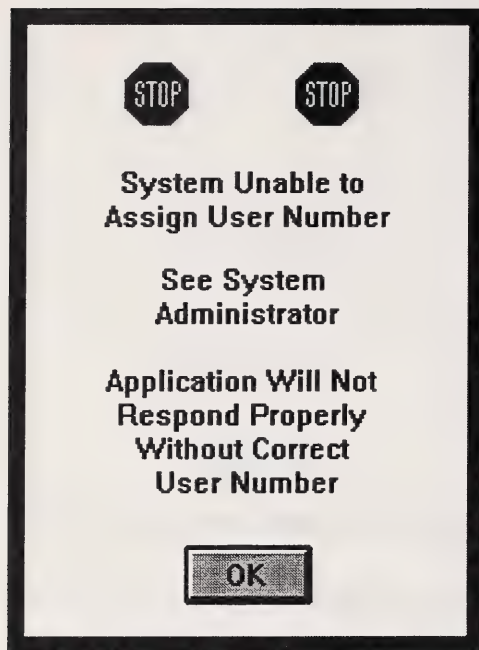




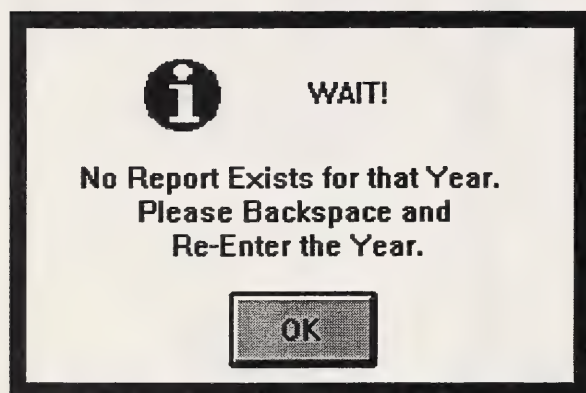
STATWRITE appears when the application is unable to write to the statistics file.



SYSERR appears when the RPC call fails during a print request.



USERERR appears when the application is unable to open the user number file or when the user number file is empty.



YEAR\_ERR appears when the incorrect year has been entered.



### C.1.2 Error Messages in the Local Error File (err.fil)

This section describes error messages that are recorded in the local error file due to communications and data errors. It outlines format and explains terminology used in the error messages.

The first line of each message will always contain three parts:

- screen or operation that error occurred in
- the remote function being called
- the type of RPC call being called

For example, if the first line of the message is: Single Query, Get Seq Header, clnt\_create, then the error occurred during a single query operation, while calling the remote function get\_seq\_header, and trying to create a client handle. A message displayed in uppercase letters reports more detailed information about the error, i.e., SEARCH ENGINE COULD NOT FIND EIN.

Additional terms incorporated into the error messages include RPC ERROR and T\_ERRNO -- numeric values which report RPC creation errors and lower level errors, respectively. Interpretation of these two errors may be accomplished by referring to the files clnt.h and xti.h (which ship with PC-NFS Toolkit) and examining values for rpc\_createerr and t\_errno.

Two other pieces of information recorded in the error file (when appropriate) are the results of the RPC call itself and the result of the RPC call returned by the search engine. These messages are indicated by the terms RPCRES (RPC result) and RES (search engine result) in the error file. Additional letters are prefixed to these terms and indicate what remote function was being called. For example, HRPCRES represents the RPC call result for the remote function get\_seq\_header (employer header). The different terms used to report RPC results and search engine RPC results include:

<b>Prefix</b>	<b>RPC Result</b>	<b>Search Engine Result</b>
A - additional matches	ARPCRES	ARES
B - blanket	BRPCRES	BRES
Br - browse	BrRPCRES	BrRES
D - detail	DRPCRES	DRES
H - header	HRPCRES	HRES
P - print	PRPCRES	PRES
S - single query	SRPCRES	SRES

The search engine will return a 0 upon success and a -1 upon failure. The rpc call will return 0 upon success and various numerical values upon failure. Again, rpc\_createerr must be interpreted (using the PC-NFS Toolkit documentation) to determine the rpc call result status.

Listed below are all of the messages that may be recorded in the local error file. They are divided into communications and data error sections and further divided into subsections which indicate what operation was being attempted when the error occurred.

## **Communications Errors**

### **Single Query Messages:**

Single Query, Get Seq Header, clnt\_create

CLIENT HANDLE IS NULL

\* = RPC ERROR

\* = T\_ERRNO

Single Query, Get Seq Header, clnt\_control

RE-TRY TIMEOUT WAS NOT SET

Single Query, Get Seq Header, clnt\_call

HRPCRES = \*\*

HRES = \*\*

\* = RPC ERROR

\* = T\_ERRNO

Single Query, Single Query, clnt\_create

CLIENT HANDLE IS NULL

\* = RPC ERROR

\* = T\_ERRNO

Single Query, Single Query, clnt\_control

RE-TRY TIMEOUT WAS NOT SET

Single Query, Single Query, clnt\_call

\*\* = SRPCRES

\*\* = SRES

\* = RPC ERROR

\* = T\_ERRNO

### **Blanket Messages:**

Query Info, Get Blanket, clnt\_create

CLIENT HANDLE IS NULL

\* = RPC ERROR

\* = T\_ERRNO

Query Info, Get Blanket, clnt\_control

RE-TRY TIMEOUT WAS NOT SET



Query Info, Get Blanket, clnt\_call  
\*\* = BRPCRES  
\*\* = BRES  
\* = RPC ERROR  
\* = T\_ERRNO

**Browse Report Messages:**

Browse, Browse Report, clnt\_create  
CLIENT HANDLE IS NULL  
\* = RPC ERROR  
\* = T\_ERRNO

Browse, Browse Report, clnt\_control  
RE-TRY TIMEOUT WAS NOT SET

Browse, Browse Report, clnt\_call  
\*\* = BrRPCRES  
\*\* = BrRES  
\* = RPC ERROR  
\* = T\_ERRNO

Browse, Get Seq Header, clnt\_create  
CLIENT HANDLE IS NULL  
\* = RPC ERROR  
\* = T\_ERRNO

Browse, Get Seq Header, clnt\_control  
RE-TRY TIMEOUT WAS NOT SET

Browse, Get Seq Header, clnt\_call  
\*\* = BrRPCRES  
\*\* = BrRES  
\* = RPC ERROR  
\* = T\_ERRNO

Browse, Get Empl Detail, clnt\_create  
CLIENT HANDLE IS NULL  
\* = RPC ERROR  
\* = T\_ERRNO

Browse, Get Empl Detail, clnt\_control  
RE-TRY TIMEOUT WAS NOT SET

Browse, Get Empl Detail, clnt\_call

\*\* = DRPCRES

\*\* = DRES

\* = RPC ERROR

\* = T\_ERRNO

#### **Print Report Messages:**

Print , Print Report, rpc\_call

\*\* = PRPCRES

\*\* = PRES

\* = RPC ERROR

\* = T\_ERRNO

#### **Query Info Messages:**

Change Header, Get Seq Header, clnt\_create

CLIENT HANDLE IS NULL

\* = RPC ERROR

\* = T\_ERRNO

Change Header, Get Seq Header, clnt\_control

RE-TRY TIMEOUT WAS NOT SET

Change Header, Get Seq Header, clnt\_call

HRPCRES = \*\*

HRES = \*\*

\* = RPC ERROR

\* = T\_ERRNO

Query Info, Add Matches, clnt\_create

CLIENT HANDLE IS NULL

\* = RPC ERROR

\* = T\_ERRNO

Query Info, Add Matches, clnt\_control

RE-TRY TIMEOUT WAS NOT SET

Query Info, Add Matches, clnt\_call

\*\* = ARPCRES

\*\* = ARES

\* = RPC ERROR

\* = T\_ERRNO



Query Info, Get Empl Detail, clnt\_create  
CLIENT HANDLE IS NULL  
\* = RPC ERROR  
\* = T\_ERRNO

Query Info, Get Empl Detail, clnt\_control  
RE-TRY TIMEOUT WAS NOT SET

Query Info, Get Empl Detail, clnt\_call  
\*\* = DRPCRES  
\*\* = DRES  
\* = RPC ERROR  
\* = T\_ERRNO

## **Data Errors**

### **Single Query Messages:**

Single Query, Get Seq Header, clnt\_call  
HRPCRES = \*\*  
HRES = \*\*  
SEARCH ENGINE COULD NOT FIND EIN

Single Query, Single Query, clnt\_call  
NO MATCHES TO THIS QUERY

### **Blanket Messages:**

Query Info, Get Blanket, clnt\_call  
\*\* = BRPCRES  
\* = BRES  
SEARCH ENGINE COULD NOT FIND SEQUENCE NUMBER

### **Browse Report Messages:**

Browse, Browse Report, clnt\_call  
BrRPCRES = \*\*  
BrRES = \*\*  
SEARCH ENGINE COULD NOT FIND MRN

Browse, Get Seq Header, clnt\_call  
BrRPCRES = \*\*  
BrRES = \*\*  
SEARCH ENGINE COULD NOT FIND EIN

**Print Report Messages:**

Print, Print Report, rpc\_call

\*\* = PRPCRES

\*\* = PRES

SEARCH ENGINE COULD NOT FIND EIN or SEQ NO.

OR REPORT CONTAINS MORE THAN 5000 EMPLOYEES

**Query Info Messages:**

Change Header, Get Seq Header, clnt\_call

HRPCRES = \*\*

HRES = \*\*

SEARCH ENGINE COULD NOT FIND SEQUENCE NUMBER

Query Info, Add Matches, clnt\_call

NO MORE MATCHES FOUND

\*\* = ARES

\* numeric value held in the variable rpc\_createerr or t\_erno

\*\* appropriate numerical value for result of RPC call or result  
returned by search engine

**C.2 File Server Applications Error Messages**

This section covers error messages generated by applications resident on the file server. These are errors that could occur during the data conversion process, the indexing process, or the searching process. They could also occur in any of the debugging programs resident on the file server.

*ERROR: Cannot create client handle*

This error can occur in the client.c debugging program. It will occur if the client handle used to access the RPC call cannot be created. A possible reason for the error may be that the file server search applications are not running.

*ERROR: Cannot modify client handle*

This error can occur in the client.c debugging program. It will occur if the client handle used to access the RPC call cannot be modified to extend the time-out.



*ERROR: Cannot open file <filename>*

This is a general error message which can occur in any of the application modules. It will occur if a file cannot be created or open (for read or write). Possible reasons for this error are if the file does not exist, or if the permissions on the file or the directory are not set correctly.

*ERROR: Cannot read detail file*

*ERROR: Cannot read query*

*ERROR: Cannot write query*

*ERROR: Cannot write to output file*

These errors occur when a read or write error is encountered. They could occur in any of the search application modules, or the data conversion modules, or the debugging applications. Possible reasons for these errors are if the permissions on the file or the directory are not set correctly.

*ERROR: Cannot register <function>*

This error could occur in any of the search application modules. It will occur if one of the RPC functions cannot be registered for use.

*ERROR: Could not complete operation*

This error could occur in the debugging application. It will occur if any of the debug choices return a value of FAIL. It will always be accompanied by a more descriptive message printed within the failed debug module.

*ERROR: Employer header not found for this EIN*

This error could occur in any of the searching applications. It will occur if the specified EIN in the request does not have an associated report.

*ERROR: Employer header not found for this Sequence*

This error could occur in any of the searching applications. It will occur if the specified EIN and sequence number in the request does not have an associated report.

*ERROR: Employer Report Too Large to Print -- See System Administrator*

This error will occur if a print report is requested for an employer report which is over 5000 records. It is determined to be an error in order to limit the size of employer reports that a user can print. If the report needs to be printed, the system administrator can print the report.

*ERROR: File inconsistency in <filename> -- 1st record not an employer header*

This error will occur during the data conversion of the COM file. It will occur because it is expecting an employer report to always begin with an employer header record. This error does not mess up the parse process, there is no need to run fix\_parse. Just re-parse this employer file once the inconsistency is fixed.

*ERROR: Illegal Employee Record in <filename>*

This error will occur during the "check data" option of the debugging application. It will occur if an employee record occurs after a final total record, or before the employer header record.

*ERROR: Invalid typeid at offset <offset>*

This error will occur during a detailed employee information request if the offset sent to the search application does not match an employee detail record.

*ERROR: No entry in parse control file for <EIN>*

This error message can occur in the fix\_parse application when an EIN or an EIN/sequence number combination is chosen to be deleted, and it is not in the parse control file.

*ERROR: No more matches available*

This error will occur during an Add Matches request if there are no more matches for the current search query. It is actually not an error message, but more an indication of the end of an operation.

*ERROR: Returned from svc\_run() in <function>*

This error message can occur during any of the searching applications. It will occur if the application returns from the RPC request wait loop.

*ERROR: Search\_record() function returned FAIL*

This error message can occur during the search single application, and will occur if there is a system error during the actual search. It would most likely occur if enough memory is not available to allocate for the search.

*ERROR: This file already exists*

This error message can occur during the debugging application. It will occur if the output file specified already exists. It is done in order to avoid wiping out important files, such as index or search statistic files.



*ERROR: Unable to Allocate Memory*

This error message can occur during the indexing application. It will occur if enough memory cannot be allocated to create and store the btree in memory. If this error occurs, simply remove all of the .idx, .dup, and .tmp files created by the index application for the EIN that the application fails on, and reindex the employer report.

*ERROR: Unidentified Record in <filename> at location <offset>*

This error message can occur during the data conversion process. It will occur if there is an unidentified record type in the COM file (if there is garbage in the file).

*ERROR: Unable to browse report*

This error message can occur during the search browse application. It will occur if a browse file for the specified EIN and sequence number does not exist.





## **D Listing of the Code**

Section D.1 contains the user interface code and section D.2 contains the search engine code. The page numbering in these sections is formatted such that each makefile, code module, include file and/or library, has its source code numbered individually.





## **D.1 User Interface Code (including utilities)**





```
// Filename: GENERIC.H
// "EAMAT42" Generated by WindowsMAKER Professional.
// Author: Laura L. Downey

//
// *****
// Do not add code here.
//
// This file is maintained by WindowsMAKER Professional.
// As you make changes in your application using WindowsMAKER Professional,
// this file is automatically updated, therefore you never modify this file.
//
//
// For more information,
// see the section "How code is generated" in the documentation.
//
// *****
//

#include "MFEDIT.H"
#include "STRUCT.H"
#include "STRING.H"
#include "GLOBAL1.H"
#include "STDIO.H"
#include "STDLIB.H"
#include "CTYPE.H"

// Give access to variables in all code modules
extern HINSTANCE hInst;
extern HWND MainhWnd;
extern UINT wHelpMessage;
extern BOOL bHelpSupport;
extern DWORD dwDialogProp;
extern BOOL b256Color;
extern HINSTANCE hBMPInst;
extern HWND hMDIClient; // Handle to client window for MDI.
extern UINT idMDIFirstChild; // ID to first child for MDI.
extern HWND MAINhDlg;
extern DLGPROC MAINlpProc;

// *****
// Variables, types and constants for controls in main window.
// *****

extern HWND hClient; // Handle to window in client area.
extern DLGPROC lpClient; // Function for window in client area.
extern UINT wBLDWindowType;

#define CLIENTSTRIP WS_MINIMIZE|WS_MAXIMIZE|WS_CAPTION|WS_BORDER|WS_DLGFRAAME|WS_SYSMENU|WS_
_POPUP|WS_THICKFRAME|DS_MODALFRAME|DS_SYSMODAL
#define TOOLBARSTRIP WS_MINIMIZE|WS_MAXIMIZE|WS_POPUP|DS_SYSMODAL

typedef struct
{
    long style;
    // MORE ...
} BLD_DLGTTEMPLATE;

typedef BLD_DLGTTEMPLATE far *LPBLD_DLGTTEMPLATE;

#define BLDTOOLBARTOP 1
#define BLDTOOLBARBOTTOM 2
#define BLDTOOLBARLEFT 3
```

```
#define BLDTOOLBARRIGHT 4
#define BLDDLGLCLIENT 5
#define BLDDLGMODAL 6
#define BLDDLGMODELESS 7
```

```
// Constants for error message strings
```

```
#define BLD_CannotRun 4000
#define BLD_CannotCreate 4001
#define BLD_CannotLoadMenu 4002
#define BLD_CannotLoadIcon 4003
#define BLD_CannotLoadBitmap 4004
#define BLD_CannotCreateWindow 4005
```

```
// User defined constant ID's
```

```
#define IDMF_Password 10005
#define IDLB_QMatch 10009
#define IDMF_QEIN 10012
#define IDMF_PEIN 10018
#define ID_BEmprHeader 10021
#define ID_QEmprHeader 10022
#define IDLB_BMatch 10023
#define ID_HDetail 10024
#define ID_EDetail 10027
#define IDPRINTED 10029
#define IDPRINTHD 10030
#define IDPRINTB 10031
#define IDBlanket 10032
#define ID_QAddMatch 10034
#define ID_BrEmprHeader 10035
#define IDLB_BrMatch 10036
#define IDMF_REIN 10039
#define IDMF_BrEIN 10044
#define ID_BrRepTot 10050
#define ID_BRepTot 10051
#define ID_QRepTot 10052
#define ID_BrAddRec 10053
#define IDPRINTTOT 10055
#define ID_CONT 10056
#define ID_Tot 10057
#define IDMF_NumCopies 10059
#define IDMF_QYear 10060
#define IDMF_QEstab 10061
#define IDMF_LName 10062
#define IDMF_FName 10063
#define IDMF_QSSN 10064
#define IDMF_BrYear 10067
#define IDMF_BrEstab 10068
#define IDMF_BrStart 10069
#define IDMF_PYear 10071
#define IDMF_PEstab 10072
#define IDMF_RYear 10073
#define IDMF_REstab 10074
#define IDCONT 10075
#define IDMF_PSeq 10076
#define ID_Change 10077
#define ID_SeqNo 10078
#define ID_TotEIN 10079
#define ID_TotRpt 10080
#define qp_year 10087
#define qp_ein 10088
#define qp_estab 10089
```

```
#define qp_lname 10090
#define qp_fname 10091
#define qp_ssn 10092
#define ID_Q1Yes 10093
#define ID_Q1No 10094
#define ID_Q2Yes 10095
#define ID_Q2No 10096
#define ID_CLOSE 10097
```

```
// WindowsMAKER automatic generated constant ID's
```

```
#define WMPDEBUG static void WMPDebugDummy() {}
```

```
// Help ID's used by functions
```

```
#if !defined(THISISBLDRC)
```

```
int PASCAL WinMain(HINSTANCE, HINSTANCE, LPSTR, int);
LRESULT CALLBACK BLDMainWndProc(HWND, UINT, WPARAM, LPARAM);
LRESULT BLDDefWindowProc(HWND, UINT, WPARAM, LPARAM);
BOOL BLDKeyTranslation(MSG *);
BOOL BLDInitApplication(HINSTANCE, HINSTANCE, int *, LPSTR);
BOOL BLDExitApplication(void);
HWND BLDCreateClientControls(char *, DLGPROC);
BOOL BLDInitMainMenu(HWND);
BOOL BLDMenuCommand(HWND, UINT, WPARAM, LPARAM);
BOOL BLDMenuHelp(HWND, UINT, WPARAM, LPARAM);
BOOL BLDRegisterClass(HINSTANCE);
HWND BLDCreateWindow(HINSTANCE);
int BLDDisplayMessage(HWND, UINT, char *, int);
BOOL BLDSwitchMenu(HWND, char *);
HMENU BLDLoadMenu(HWND, char *, HMENU *);
BOOL BLDDrawBitmap(LPDRAWITEMSTRUCT, char *, BOOL);
BOOL BLDDrawIcon(LPDRAWITEMSTRUCT, char *);
void BLDMoveWindow(HWND, int, int, int, int, BOOL);
BOOL BLDSendMDIMessage(HWND, UINT);
BOOL BLDDrawBkgndIcon(HWND, PAINTSTRUCT *, char *, int);
BOOL BLDDrawBkgndBitmap(HWND, PAINTSTRUCT *, char *, int, BOOL, BOOL);
BOOL BLDDrawAutoState(LPDRAWITEMSTRUCT, char *, BOOL, BOOL);
BOOL BLDDrawStateBitmap(LPDRAWITEMSTRUCT, char *, char *, char *, char *, BOOL);
BOOL BLDDrawStateIcon(LPDRAWITEMSTRUCT, char *, char *, char *, char *);
void BLDGetDlgScrolled(HWND, int *, int *);
void BLDSetDlgScrolled(HWND, int, int);
void BLDFindCtrlsRightBottom(HWND, int *, int *);
void BLDCalcScrollRanges(HWND, int *, int *, int, int, int, int);
BOOL BLDScrollDlg(HWND, UINT, int, int, int, int, int, int, int, int, BOOL);
BOOL BLDExitScrollDlg(HWND);
BOOL BLDSizedlg(HWND, int, int);
HBITMAP BLDLoadBitmap(HANDLE, char *);
HWND BLDCreateClientDlg(char *, HWND, UINT, DLGPROC, int, BOOL);
LRESULT BLDSizeToolBars(HWND, UINT, int, int, int, BOOL);
void BLDCalcToolbarFrame(HWND, int *, int *, int *, int *);
void BLDMoveDlgClient(HWND, HWND);
void BLDSetClientFocus(HWND hWnd);
void BLDClientMove(HWND hWnd);
BOOL BLDInitCtrlFont(HWND, int, int, int, int, int, int, int, BYTE, BYTE, BYTE, BYTE, BYTE, BYTE,
    BYTE, BYTE, char *);
BOOL BLDExitCtrlFont(HWND, int);
BOOL BLDInitSolidBrush(HWND, COLORREF);
BOOL BLDInitPatternBrush(HWND, char *);
BOOL BLDExitBrush(HWND);
```



```
HBRUSH BLDctlColorStockBrush(HWND,int);
HBRUSH BLDctlColorPropBrush(HWND);
HBRUSH BLDctlColorDefaultBrush(HWND);
HBRUSH BLDctlColorBrushSetOrg(HWND,HDC);
BOOL BLDCheckFlHelpKey(BOOL);
void BLDHelpTranslation(MSG *);
void BLDShowHelp(HWND,UINT,DWORD);
BOOL BLDHelpFilter(HWND,UINT,WPARAM,LPARAM,DWORD,LPARAM *,BOOL);
BOOL BLDGetHelpFileName(char *);
LRESULT BLDDefWindowProcMsg(HWND,UINT,WPARAM,LPARAM);
BOOL BLDMainRegClass(HINSTANCE);
BOOL BLDMainRegClassDef(HINSTANCE);
BOOL BLDMainExitClass(void);
BOOL BLDMainExitClassDef(void);
HWND BLDMainCreateWnd(void);
HWND BLDMainCreateWndDef(void);
BOOL BLDWndMsgFilter(HWND,UINT,WPARAM,LPARAM,DWORD,LRESULT *);
BOOL BLDDlgMsgFilter(HWND,UINT,WPARAM,LPARAM,int,DWORD,BOOL *);
BOOL BLDAddClientDlg(HWND,DLGPROC);
BOOL BLDRemoveClientDlg(HWND);
BOOL BLDIsClientDlgDialogMessage(MSG *);
BOOL BLDDrawItem(HWND,LPDRAWITEMSTRUCT);
BOOL BLDBitmapToScreen(HDC,char *,int,int,int,int,DWORD,BOOL);

void BLDMoveDlgClientDef(HWND,HWND);
int BLDDisplayMessageDef(HWND,UINT,char *,int);
BOOL BLDDrawBitmapDef(LPDRAWITEMSTRUCT,char *,BOOL);
BOOL BLDDrawIconDef(LPDRAWITEMSTRUCT,char *);
void BLDMoveWindowDef(HWND,int,int,int,int,BOOL);
BOOL BLDSendMDIMessageDef(HWND,UINT);
BOOL BLDDrawBkgndIconDef(HWND,PAINTSTRUCT *,char *,int);
BOOL BLDDrawBkgndBitmapDef(HWND,PAINTSTRUCT *,char *,int,BOOL,BOOL);
BOOL BLDDrawAutoStateDef(LPDRAWITEMSTRUCT,char *,BOOL,BOOL);
BOOL BLDDrawStateBitmapDef(LPDRAWITEMSTRUCT,char *,char *,char *,char *,BOOL);
BOOL BLDDrawStateIconDef(LPDRAWITEMSTRUCT,char *,char *,char *,char *);
void BLDGetDlgScrolledDef(HWND,int *,int *);
void BLDSetDlgScrolledDef(HWND,int,int);
void BLDFindCtrlsRightBottomDef(HWND,int *,int *);
void BLDCalcScrollRangesDef(HWND,int *,int *,int,int,int,int);
BOOL BLDScrollDlgDef(HWND,UINT,int,int,int,int,int,int,int,BOOL);
BOOL BLDExitScrollDlgDef(HWND);
BOOL BLDSizeDlgDef(HWND,int,int);
HWND BLDCreateClientDlgDef(char *,HWND,UINT,DLGPROC,int,BOOL);
LRESULT BLDSizeToolBarsDef(HWND,UINT,int,int,int,BOOL);
void BLDCalcToolbarFrameDef(HWND,int *,int *,int *,int *);
void BLDSetClientFocusDef(HWND hWnd);
void BLDClientMoveDef(HWND hWnd);
BOOL BLDInitCtrlFontDef(HWND,int,int,int,int,int,int, BYTE, BYTE, BYTE, BYTE, BYTE, BY
TE, BYTE, BYTE, char *);
BOOL BLDExitCtrlFontDef(HWND,int);
BOOL BLDInitSolidBrushDef(HWND,COLORREF);
BOOL BLDInitPatternBrushDef(HWND,char *);
BOOL BLDExitBrushDef(HWND);
HBRUSH BLDctlColorStockBrushDef(HWND,int);
HBRUSH BLDctlColorPropBrushDef(HWND);
HBRUSH BLDctlColorDefaultBrushDef(HWND);
HBRUSH BLDctlColorBrushSetOrgDef(HWND,HDC);
BOOL BLDCheckFlHelpKeyDef(BOOL);
void BLDHelpTranslationDef(MSG *);
void BLDShowHelpDef(HWND,UINT,DWORD);
BOOL BLDHelpFilterDef(HWND,UINT,WPARAM,LPARAM,DWORD,LPARAM *,BOOL);
LRESULT BLDDefWindowProcMsgDef(HWND,UINT,WPARAM,LPARAM);
BOOL BLDAddClientDlgDef(HWND,DLGPROC);
```

```
BOOL BLDRemoveClientDlgDef (HWND);
BOOL BLDIsClientDlgDialogMessageDef (MSG *);
HWND BLDCreateClientControlsDef (char *, DLGPROC);
BOOL BLDDrawItemDef (HWND, LPDRAWITEMSTRUCT);
static BOOL BLDMoveTo (HDC, int, int);
static BOOL BLDDrawFrame (HDC, int, int, int, int, int, BOOL);
BOOL BLDBitmapToScreenDef (HDC, char *, int, int, int, int, int, DWORD, BOOL);
HBRUSH BLDGetGlobalBrushDef (HWND hCtrl, HDC hDC);

int BLD_QUERYDlgFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_QUERYDlgFuncDef (HWND hWnd, char *szDlgName);
BOOL CALLBACK BLD_QUERYDlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
BOOL BLD_QUERYDlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
HWND BLD_MAINCFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
HWND BLD_MAINCFuncDef (HWND hWnd, char *szDlgName, UINT message);
BOOL CALLBACK BLD_MAINCProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
BOOL BLD_MAINDlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_PrintDlgFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_PrintDlgFuncDef (HWND hWnd, char *szDlgName);
BOOL CALLBACK BLD_PrintDlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
BOOL BLD_PrintDlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_OKDlgFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_OKDlgFuncDef (HWND hWnd, char *szDlgName);
BOOL CALLBACK BLD_OKDlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
BOOL BLD_OKDlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
BOOL BLD_ApplicationAppInit (HINSTANCE hInst, HINSTANCE hPrev, int *pCmdShow, LPSTR lpCmd);
int BLD_FunctionDlgFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_FunctionDlgFuncDef (HWND hWnd, char *szDlgName);
BOOL CALLBACK BLD_FunctionDlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
BOOL BLD_FunctionDlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_Function2DlgFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_Function2DlgFuncDef (HWND hWnd, char *szDlgName);
BOOL CALLBACK BLD_Function2DlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
BOOL BLD_Function2DlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_Function6DlgFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_Function6DlgFuncDef (HWND hWnd, char *szDlgName);
BOOL CALLBACK BLD_Function6DlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
BOOL BLD_Function6DlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
BOOL BLD_QuitFuncUDCFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_HeaderDetailDlgFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_HeaderDetailDlgFuncDef (HWND hWnd, char *szDlgName);
BOOL CALLBACK BLD_HeaderDetailDlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
BOOL BLD_HeaderDetailDlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_EmployeeDetailDlgFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_EmployeeDetailDlgFuncDef (HWND hWnd, char *szDlgName);
BOOL CALLBACK BLD_EmployeeDetailDlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
);
BOOL BLD_EmployeeDetailDlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_BrowseReportDlgFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_BrowseReportDlgFuncDef (HWND hWnd, char *szDlgName);
BOOL CALLBACK BLD_BrowseReportDlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
BOOL BLD_BrowseReportDlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_ReportStatisticsDlgFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_ReportStatisticsDlgFuncDef (HWND hWnd, char *szDlgName);
BOOL CALLBACK BLD_ReportStatisticsDlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
am);
BOOL BLD_ReportStatisticsDlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_ReportDlgFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_ReportDlgFuncDef (HWND hWnd, char *szDlgName);
BOOL CALLBACK BLD_ReportDlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
BOOL BLD_ReportDlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_BrowseEntryDlgFunc (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);
int BLD_BrowseEntryDlgFuncDef (HWND hWnd, char *szDlgName);
BOOL CALLBACK BLD_BrowseEntryDlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);
```



```
BOOL BLD_BrowseEntryDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_ReportTotalsDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_ReportTotalsDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_ReportTotalsDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_ReportTotalsDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_EINerrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_EINerrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_EINerrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_EINerrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_QueryMessageDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_QueryMessageDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_QueryMessageDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_QueryMessageDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_NMSGDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_NMSGDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_NMSGDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_NMSGDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_QueryErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_QueryErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_QueryErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_QueryErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_BlanketErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_BlanketErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_BlanketErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_BlanketErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_PWErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_PWErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_PWErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_PWErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_Year_ErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_Year_ErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_Year_ErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_Year_ErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_SysErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_SysErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_SysErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_SysErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_BlankErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_BlankErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_BlankErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_BlankErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_MissingFileDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_MissingFileDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_MissingFileDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_MissingFileDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_QueryTxtDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_QueryTxtDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_QueryTxtDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_QueryTxtDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_ErrorFileDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_ErrorFileDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_ErrorFileDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_ErrorFileDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_NULLPtrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_NULLPtrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_NULLPtrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_NULLPtrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_DataErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_DataErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_DataErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_DataErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_DFileErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_DFileErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_DFileErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
```



```
BOOL BLD_DFileErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_NoMoreMatchesDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_NoMoreMatchesDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_NoMoreMatchesDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
;
BOOL BLD_NoMoreMatchesDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_SequenceDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_SequenceDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_SequenceDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_SequenceDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_MRNErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_MRNErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_MRNErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_MRNErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_MatchErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_MatchErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_MatchErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_MatchErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_PrintBlanketDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_PrintBlanketDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_PrintBlanketDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_PrintBlanketDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_PrintEmpDetailDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_PrintEmpDetailDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_PrintEmpDetailDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
);
BOOL BLD_PrintEmpDetailDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_PrintHeaderDetailDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_PrintHeaderDetailDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_PrintHeaderDetailDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
);
BOOL BLD_PrintHeaderDetailDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_Function5DlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_Function5DlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_Function5DlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_Function5DlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_TotNowPrintDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_TotNowPrintDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_TotNowPrintDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_TotNowPrintDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_GetNumCopyDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_GetNumCopyDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_GetNumCopyDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_GetNumCopyDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_qparamDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_qparamDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_qparamDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_qparamDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_EINorSeqErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_EINorSeqErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_EINorSeqErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_EINorSeqErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_SeqErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_SeqErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_SeqErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_SeqErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_HFileDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_HFileDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_HFileDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_HFileDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_UserNumErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_UserNumErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_UserNumErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_UserNumErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
```

```
int BLD_questDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_questDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_questDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_questDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_StatOpenDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_StatOpenDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_StatOpenDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_StatOpenDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_StatEmptyDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_StatEmptyDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_StatEmptyDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_StatEmptyDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_StatWOpenDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_StatWOpenDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_StatWOpenDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_StatWOpenDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_StatWriteErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam);
int BLD_StatWriteErrDlgFuncDef(HWND hWnd,char *szDlgName);
BOOL CALLBACK BLD_StatWriteErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
BOOL BLD_StatWriteErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam);
```

```
#endif
```

```
// *****
// ERROR MESSAGE HANDLING (Definitions can be overruled.)
// *****
```

```
#ifndef BLDMAINCAPTION
#define BLDMAINCAPTION "EAMATE V4.2"
#endif
```

```
#ifndef BLDLOADERROR
#define BLDLOADERROR "Cannot load string."
#endif
```

```
// WindowsMAKER color definitions
```

```
#define BLD_BLACK 0X00000000L
#define BLD_WHITE 0X00FFFFFFL
#define BLD_GRAY 0X007F7F7FL
#define BLD_LTGRAY 0X00C0C0C0L
```

```
// WindowsMAKER definitions
```

```
#define BLD_MAXPATH 256
```

```
// WindowsMAKER global dialog box properties
```

```
#define BLDGRAY_DIALOGBOX 0x00000001L
#define BLDGRAY_BUTTON 0x00000002L
#define BLDGRAY_COMBOBOX 0x00000004L
#define BLDGRAY_LISTBOX 0x00000008L
#define BLDGRAY_EDIT 0x00000010L
#define BLDGRAY_SCROLLBAR 0x00000020L
#define BLDGRAY_TEXT 0x00000040L
```



```
/** Global.h **/
```

```
/** by Laura L. Downey **/
```

```
/** 10/13/92 **/
```

```
/** This file defines global variables to be used by the eamate **/
```

```
/** prototype custom code; to be included in usercode.c **/
```

```
/** all variables included here are defined externally in global.h **/
```

```

char QYear[5],          /** Info from Query Dialog Box **/
  QEIN[11],             /** user input from child window IDMF_Qyear **/
  QEstab[5],            /** user input from child window IDMF_QEIN **/
  LName[16],            /** user input from child window IDMF_QEstab **/
  FName[13],            /** user input from child window IDMF_LName **/
  QSSN[12],             /** user input from child window IDMF_FName **/
                        /** user input from child window IDMF_QSSN **/

  BrYear[5],            /** Info from NewBrowse Dialog Box **/
  BrEIN[11],            /** user input from child window IDMF_BrYear **/
  BrEstab[5],           /** user input from child window IDMF_BrEIN **/
  BrStart[12],          /** user input from child window IDMF_BrEstab **/
                        /** user input from child window IDMF_BrStart **/

  PW[10],               /** Info from Print Dialog Box **/
  PYear[5],             /** user input from child window IDMF_Password **/
  PEIN[11],             /** user input from child window IDMF_PYear **/
  PEstab[5],            /** user input from child window IDMF_PEIN **/
  PSeq[4],              /** user input from child window IDMF_PEstab **/
                        /** user input from child window IDMF_PSeq **/

  RYear[5],             /** Info from Report Dialog Box **/
  REIN[11],             /** user input from child window IDMF_RYear **/
  REstab[5],            /** user input from child window IDMF_REIN **/
                        /** user input from child window IDMF_REstab **/

  cnum[6];              /** Info from NCopy Dialog Box **/
                        /** user input from child window IDMF_NumCopies **/

int offcount;           /** # of browse offsets or records that have been read **/
int USER;              /** user number, read from file usernum.fil **/
long int error;         /** used for debugging purposes **/

char sequence[4];       /** holds sequence number of employer report **/
                        /** may be input from child window IDMF_SeqNo **/

/***** STRING VARIABLES *****/
char EDetailString[700]; /** added 7/6/92 - holds employee detail string for display **/
char HeaderString[300];  /** added 7/15/92 - holds shortened employer header string for display **/
char HDetailString[400]; /** added 7/15/92 - holds detailed employer header info for display **/
char PHDetailString[400]; /** added 7/15/92 - hold detailed employer header info to be sent to printer **/
char PEDetailString[700]; /** added 7/15/92 - holds detailed employee info to be sent to printer **/
char TotalString[700];   /** added 8/19/92 - holds totals info for display **/
char PTotalString[700];  /** added 8/19/92 - holds totals info for printing **/
char PrintTotString[700]; /** added 8/21/92 - includes title, header detail and report totals to be sent to printer **/
char PrintEDetail[1100]; /** added 8/21/92 - holds title, header detail and employee detail for printing **/

```

```

/***** STRUCTURE VARIABLES *****/
struct W2EmpInfo Blanket[30];    /** added 11/30/92 by LLD to hold blanket records **/
struct W2Browse Browse;         /** added 11/16/92 by LLD to hold employee browse
                                info to be displayed in listboxes **/
struct W2EmpInfo EDetail;       /** added 11/16/92 by LLD to hold dummy employee
                                detail data **/
struct W2EmprInfo CurrEmprInfo; /** added 9/30/92 by LLD **/
                                /** holds empr header, final totals, **/
                                /** and cumein (if it exists) re current **/
                                /** employer being searched **/
                                /** explicitly defined in global.h which is **/
                                /** included in usercode.c **/

/** separatorstring used when printing blanket report to separate employee records **/
char separatorstring [] = "\n-----\n";
-----\n";
```



```
/** Global1.h **/
```

```
/** by Laura L. Downey **/
```

```
/** 10/13/92 **/
```

```
/** This file defines global variables listed in global.h and main.h  
as external; to be used by the eamate prototype custom code **/
```

```
extern char QYear[5],  
    QEIN[11],  
    QEstab[5],  
    LName[16],  
    FName[13],  
    QSSN[12],
```

```
/** Info from Query Dialog Box **/
```

```
/** user input from child window IDMF_Qyear **/  
/** user input from child window IDMF_QEIN **/  
/** user input from child window IDMF_QEstab **/  
/** user input from child window IDMF_LName **/  
/** user input from child window IDMF_FName **/  
/** user input from child window IDMF_QSSN **/
```

```
    BrYear[5],  
    BrEIN[11],  
    BrEstab[5],  
    BrStart[12],
```

```
/** Info from NewBrowse Dialog Box **/
```

```
/** user input from child window IDMF_BrYear **/  
/** user input from child window IDMF_BrEIN **/  
/** user input from child window IDMF_BrEstab **/  
/** user input from child window IDMF_BrStart **/
```

```
    PW[10],  
    PYear[5],  
    PEIN[11],  
    PEstab[5],  
    PSeq[4],
```

```
/** Info from Print Dialog Box **/
```

```
/** user input from child window IDMF_Password **/  
/** user input from child window IDMF_PYear **/  
/** user input from child window IDMF_PEIN **/  
/** user input from child window IDMF_PEstab **/  
/** user input from child window IDMF_PSeq **/
```

```
    RYear[5],  
    REIN[11],  
    REstab[5],
```

```
/** Info from Report Dialog Box **/
```

```
/** user input from child window IDMF_RYear **/  
/** user input from child window IDMF_REIN **/  
/** user input from child window IDMF_REstab **/
```

```
    cnum[6];
```

```
/** Info from NCopy Dialig Box **/
```

```
/** user input from child window IDMF_NumCopies **/
```

```
extern int offcount;  
read **/
```

```
/** # of browse offsets or records that have been
```

```
extern int USER;
```

```
/** user number, read from file usernum.fil **/
```

```
extern long int error;
```

```
/** used for debugging purposes**/
```

```
extern char sequence[4];
```

```
/** holds sequence number of employer report **/  
/** may be input from child window IDMF_SeqNo **/
```

```
/****** STRING VARIABLES *****/
```

```
extern char EDetailString[700];  
extern char HeaderString[300];  
r
```

```
/**added 7/6/92 - hold employee detail string**/  
/** added 7/15/92 - holds shortened employer head
```

```
extern char HDetailString[400];  
or
```

```
info for display**/  
/** added 7/15/92 - holds detailed employee info f
```

```
extern char PHDetailString[400];
```

```
display**/  
/** added 7/15/92 - holds detailed employer header  
info to be sent to printer **/
```

```
extern char PEDetailString[700];  
o be
```

```
/** added 7/15/92 - holds detailed employee info t  
sent to printer **/
```

```
extern char TotalString[700];  
**/
```

```
/** added 8/19/92 - holds totals info for display
```

```
extern char PTotalString[700];
```

```
/** added 8/19/92 - holds totals info for printing
```

```

**/
extern char PrintTotString[700];          /** added 8/21/92 - holds title, header detail and
                                          report totals for printing **/
extern char PrintEDetail[1100];          /** added 8/21/92 - holds title, header detail and
                                          employee detail for printing **/

/***** CREDITS IN MAIN WINDOW VARIABLES *****/
**/
extern FARPROC lpProc;                   /** added 6/15/92 by LLD for use in **/
extern HWND AbouthWnd;                   /** displaying credits in main window **/
                                          /** used in eamate.c WINMAIN procedure **/
                                          /** explicitly declared in main.h which is **/
                                          /** included in eamate.c **/

/***** STRUCTURE VARIABLES *****/
extern struct W2EmpInfo Blanket[30];      /** added 11/30/92 by LLD to hold blanket records
**/

extern struct W2Browse Browse;           /** added 11/16/92 by LLD to hold employee browse
                                          info to be displayed in listboxes **/

extern struct W2EmpInfo EDetail;          /** added 11/16/92 by LLD to hold dummy employee
                                          detail data **/

extern struct W2EmprInfo CurrEmprInfo;    /** added 9/30/92 by LLD **/
                                          /** holds empr header, final totals, **/
                                          /** and cumein (if it exists) re current **/
                                          /** employer being searched **/
                                          /** explicity defined in global.h which is **/
                                          /** included in usercode.c **/

extern char separatorstring[77];          /** used to separate employee records when
                                          blanket report is printed **/

/*****USER-DEFINED FUNCTIONS*****/
void CopyBlanket_EDetail(int);            /** copies current blanket detail into EDetail
so CreateEDetail() can be utilized - added 11/30/
92 **/

void CreateHeaderString(void);             /** creates browse empr header info **/
                                          /** added 6/19/92 by LLD **/

void CreateHDetailString(void);            /** added 07/01/92 by LLD - creates header detail
display string & print string**/

void CreateEDetail(void);                  /** added 7/6/92 by LLD - creates a detail string
from
list box selection if nothing entered by user
for display and a print string**/

void CreateTotalString(void);              /** added 8/19/92 by LLD - creates total string fo
r
display and printing **/

void CreatePrintTotString(void);           /** added 8/21/92 by LLD - concatenates title,
empr header, separator string, and report tota
ls **/

void CreatePrintEDetail(void);             /** added 8/21/92 by LLD - concatenates title,
empr header, & separator string, and
employee detail **/

```



```
/** NOTE: detail strings and print detail strings were created because the tabs on
the screen are different than the tabs on the printer (printer is a mono font) */

void CreateTitlePage(char[]);           /** added 7/17/92 by LLD - creates title p
age string                             for blanket report */

void fill_aggregate(void);              //added 12/6/93 by LLD, fills ex_stat aggr
agate                                  //structure

void fill_current(void);                //added 12/6/93 by LLD, fills in_stat curr
ent                                   //structure

void init_aggregate(HWND,UINT,WPARAM, LONG); //added 12/6/93 by LLD, initializes ex_sta
t                                     //aggregate structure and displays error b
oxes                                 //when appropriate

void init_current(void);                //added 12/6/93 by LLD, initializes in_sta
t                                     //current structure with all zeroes

void write_aggregate(HWND,UINT,WPARAM, LONG); //added 12/6/93 by LLD, writes aggregate
//structure to stat#.fil

BOOL FAR PASCAL AbortProc(HWND, short); //abort message function for print protect
ion

HDC GetPrinterDC(void);                 /** added 7/13/92 by LLD - creates printer
device                               context */
```

```
/** MAIN.H **/
```

```
/** 6/16/92 **/
```

```
/** by Laura L. Downey **/
```

```
/** to be included in eamate.c **/
```

```
/** defines variables to be used to insert credits in client **/
```

```
/** area of main window in eamate prototype **/
```

```
/** these are defined externally in global1.h**/
```

```
FARPROC lpProc;
```

```
HWND AbouthWnd;
```

```
// defined MAX values used by Magic Fields
```

```
#define MFMAXPATH 140
#define MFMAXCLASS 20
#define MFMAXNAME 14
#define MFMAXPICTURE 80
```

```
/* DIALOG BOX PROPERTIES USED BY Magic Fields */
/*****
```

```
#define MFCTRL_MFEDIT 0
#define MFCTRL_COMBOBOX 1
#define MFCTRL_EDIT 2
#define MFCTRL_LISTBOX 3
#define MFCTRL_SCROLLBAR 4
#define MFCTRL_STATIC 5
#define MFCTRL_GRAYFRAME 6
#define MFCTRL_BUTTON 7
#define MFCTRL_PUSHBUTTON 8
#define MFCTRL_GROUPBOX 9
#define MFCTRL_RADIOBUTTON 10
#define MFCTRL_CHECKBOX 11
```

```
#define MFCTRL_MAX 20
```

```
#define MFCBIT_GRAYCONTROL 0x0001
#define MFCBIT_NEXTLOOKUP 0x0002
#define MFCBIT_NEXTLOOKDOWN 0x0004
#define MFCBIT_DLGF1HELP 0x0001
#define MFCBIT_DLGNEXTLOOK 0x0002
```

```
/* DATA TYPES & STRUCTURES USED BY Magic Fields */
/*****
```

```
typedef struct tagMFDATE
{
    int iYear;
    int iMonth;
    int iDay;
}MFDATE;
typedef MFDATE far *LPMFDATE;
```

```
typedef struct tagMFTIME
{
    int iHour;
    int iMinute;
    int iSecond;
}MFTIME;
typedef MFTIME far *LPMFTIME;
```

```
typedef struct tagMFNUMBER
{
    BOOL bNeg;
    long lNumber;
    int iDecInitZeros;
    long lDecimal;
}MFNUMBER;
typedef MFNUMBER far *LPMFNUMBER;
```



```
typedef struct tagMFDLLINFO
{
    int      iVersion;
    BOOL     bRuntime;
    char     szPath[MFMAXPATH+1];
    char     szClass[MFMAXCLASS+1];
}MFDLLINFO;
typedef MFDLLINFO far *LPMFDLLINFO;

typedef struct tagMFFIELDINFO
{
    int      iFieldType;
    char     szValidationName[MFMAXNAME+1];
    char     szColorName[MFMAXNAME+1];
    char     szHelpName[MFMAXNAME+1];
    char     szPicture[MFMAXPICTURE+1];
    BOOL     bRangeChecking;
    BOOL     bTemplateValues;
    BOOL     bGlobalErrorHandling;
    BOOL     bGlobalInputMode;
    BOOL     bGlobalCustomProcessing;
    BOOL     bGlobalColor;
    char     szSetupFile[MFMAXPATH+1];
}MFFIELDINFO;
typedef MFFIELDINFO far *LPMFFIELDINFO;

typedef struct tagMFNOTIFICATION
{
    HWND     hField;
    WORD     message;
    WORD     wParam;
    LONG     lParam;
    DWORD    dwReturn;
    BOOL     bProcessed;
    WORD     wValidationType;
    WORD     wValidationStatus;
}MFNOTIFICATION;
typedef MFNOTIFICATION far *LPMFNOTIFICATION;

typedef struct tagMFDLGPROPERTY
{
    DWORD    dwSetup;
    DWORD    dwControl[MFCtrl_MAX];
}MFDLGPROPERTY;
typedef MFDLGPROPERTY far *LPMFDLGPROPERTY;

/*****
/* NOTIFICATION MESSAGES FROM Magic Fields */
*****/

#define MFD_NOTIFY WM_USER+333

// SUB MESSAGES
#define MFN_TESTMODE 1
#define MFN_ONLINEHELP 2
#define MFN_MENUHELP 3

#define MFN_VALIDATE 10
#define MFN_POSTVALIDATE 11

#define MFN_PROCESSMESSAGE 20
```

```
#define MFN_PROCESSHELP      21
#define MFN_PROCESSERROR    22
```

```
/* *****
/*  FORMAT TYPES used by Magic Fields
/* *****
```

```
#define MFTYPE_Custom      0
#define MFTYPE_Date       1
#define MFTYPE_Time       2
#define MFTYPE_Currency   3
#define MFTYPE_Number     4
#define MFTYPE_Integer    5
```

```
/* *****
/*  VALIDATION TYPES used by Magic Fields
/* *****
```

```
#define VTYPE_F2          1
#define VTYPE_FOCUS      2
#define VTYPE_SPECIAL    3
#define VTYPE_MESSAGE    4
#define VTYPE_CR         5
#define VTYPE_CHAR       6
#define VTYPE_SETTEXT    7
```

```
/* *****
/*  ERROR RETURNS USED BY Magic Fields
/* *****
```

```
// ERRORS BELOW 100 is internal Magic Fields error
```

```
// ERROR RETURNS FOR Magic Fields API FUNCTIONS
```

```
#define MFERR_NOPARENTWINDOW 100
#define MFERR_NOFIELD        101
#define MFERR_NOTMFIELD      102
#define MFERR_ILLEGALFIELDTYPE 103
#define MFERR_SETTEXTSTOPPED 104
#define MFERR_ILLEGALDATA    105
#define MFERR_NOTFIXEDSTRING 106
#define MFERR_TOOLONGNETSTRING 107
#define MFERR_TOOSHORTNETSTRING 108
#define MFERR_TOOSHORTSTRING 109
#define MFERR_TOOLONGSTRING  110
```

```
#define MFERR_EMPTYFIELD    120
#define MFERR_OVERFLOW      121
#define MFERR_CHARAFTERNUMBER 122
```

```
// ERRORS BETWEEN 200 and 999 is Magic Fields format errors
```

```
// ERRORS HIGHER THAN 1000 is Magic Fields validation errors
```

```
// COMMON VALIDATION ERRORS
```

```
#define VALERR_NOSYSTEMTIME 1000
#define VALERR_NEEDMORE     1001
#define VALERR_ILLEGALSEPARATOR 1002
#define VALERR_PICTUREENDSTRING 1003
#define VALERR_PICTUREENDCHAR  1004
#define VALERR_CHARACTER       1005
```

```
#define VALERR_REFORMATERROR 1006
#define VALERR_STRING 1007
#define VALERR_EMPTYFIELD 1008
#define VALERR_ILLEGALFORMAT 1009

// DATE VALIDATION ERRORS
#define VALERR_ILLEGALDAY 1050
#define VALERR_ILLEGALMONTH 1051
#define VALERR_ILLEGALYEAR 1052

// TIME VALIDATION ERRORS
#define VALERR_ILLEGALHOUR 1060
#define VALERR_ILLEGALMINUTE 1061
#define VALERR_ILLEGALSECOND 1062
#define VALERR_ILLEGALAMPM 1063

// VALUE ERRORS - ILLEGAL & OUT RANGE ERRORS
#define VALERR_ILLEGALDATE 1100
#define VALERR_SMALLDATE 1101
#define VALERR_LARGEDATE 1102

#define VALERR_ILLEGALTIME 1110
#define VALERR_SMALLTIME 1111
#define VALERR_LARGETIME 1112

#define VALERR_ILLEGALCURRENCY 1120
#define VALERR_SMALLCURRENCY 1121
#define VALERR_LARGE CURRENCY 1122

#define VALERR_ILLEGALNUMBER 1130
#define VALERR_SMALLNUMBER 1131
#define VALERR_LARGENUMBER 1132

#define VALERR_ILLEGALINTEGER 1140
#define VALERR_SMALLINTEGER 1141
#define VALERR_LARGEINTEGER 1142

#define VALERR_ILLEGALPICTURE 1150
#define VALERR_SMALLPICTURE 1151
#define VALERR_LARGEPICTURE 1152

#define VALERR_DECIMALOVERFLOW 1160
#define VALERR_NUMBEROVERFLOW 1161

// CUSTOM ERRORS
#define VALERR_CUSTOMFIRST 1200
#define VALERR_CUSTOMLAST 1299

/*****
/* LIBRARY API FUNCTIONS IN Magic Fields */
*****/

#ifdef __cplusplus
extern "C" {
#endif

WORD FAR PASCAL MfInitMFEDIT(void);
BOOL FAR PASCAL MfGetDLLInfo(LPMFDLLINFO lpInfo);
BOOL FAR PASCAL MfSetSetupFile(HANDLE hInst,LPSTR lpPath);
BOOL FAR PASCAL MfIsMagicMessage(HWND hDlg,unsigned message,WORD wParam,LONG lParam);
BOOL FAR PASCAL MfSetDlgBoxProperty(HWND hDlg,LPMFDLGPROPERTY lpFrame);

BOOL FAR PASCAL MfGetCurrentDate(LPMFDATE lpDate);
```



```
BOOL FAR PASCAL MfGetCurrentTime(LPMFTIME lpTime);

BOOL FAR PASCAL MfIsDateLegal(LPMFDATE lpDate);
BOOL FAR PASCAL MfIsTimeLegal(LPMFTIME lpTime);
BOOL FAR PASCAL MfIsNumberLegal(LPMFNUMBER lpNumber);

int FAR PASCAL MfCompareDate(LPMFDATE lpDate1,LPMFDATE lpDate2);
int FAR PASCAL MfCompareTime(LPMFTIME lpTime1,LPMFTIME lpTime2);
int FAR PASCAL MfCompareNumber(LPMFNUMBER lpNumber1,LPMFNUMBER lpNumber2);

BOOL FAR PASCAL MfIsLeapYear(int iYear);

BOOL FAR PASCAL MfDateToString(LPMFDATE lpDate,LPSTR lpStr,int max);
BOOL FAR PASCAL MfTimeToString(LPMFTIME lpTime,LPSTR lpStr,int max);
BOOL FAR PASCAL MfCurrencyToString(LPMFNUMBER lpNumber,LPSTR lpStr,int max);
BOOL FAR PASCAL MfNumberToString(LPMFNUMBER lpNumber,LPSTR lpStr,int max);

BOOL FAR PASCAL MfIsFieldValid(HWND hDlg,int nField,LPINT lpError);
BOOL FAR PASCAL MfGetFieldInfo(HWND hDlg,int nField,LPMFFIELDINFO lpInfo,LPINT lpError);

BOOL FAR PASCAL MfGetFieldDate(HWND hDlg,int nField,LPMFDATE lpDate,LPINT lpError);
BOOL FAR PASCAL MfGetFieldTime(HWND hDlg,int nField,LPMFTIME lpTime,LPINT lpError);
BOOL FAR PASCAL MfGetFieldCurrency(HWND hDlg,int nField,LPMFNUMBER lpNumber,LPINT lpError)
;
BOOL FAR PASCAL MfGetFieldNumber(HWND hDlg,int nField,LPMFNUMBER lpNumber,LPINT lpError);
BOOL FAR PASCAL MfGetFieldLong(HWND hDlg,int nField,LPLONG lpLong,LPINT lpError);
BOOL FAR PASCAL MfGetFieldNetString(HWND hDlg,int nField,LPSTR lpNet,int max,LPINT lpError
);

BOOL FAR PASCAL MfSetFieldDate(HWND hDlg,int nField,LPMFDATE lpDate,LPINT lpError);
BOOL FAR PASCAL MfSetFieldTime(HWND hDlg,int nField,LPMFTIME lpTime,LPINT lpError);
BOOL FAR PASCAL MfSetFieldCurrency(HWND hDlg,int nField,LPMFNUMBER lpNumber,LPINT lpError)
;
BOOL FAR PASCAL MfSetFieldNumber(HWND hDlg,int nField,LPMFNUMBER lpNumber,LPINT lpError);
BOOL FAR PASCAL MfSetFieldLong(HWND hDlg,int nField,LPLONG lpLong,LPINT lpError);
BOOL FAR PASCAL MfSetFieldNetString(HWND hDlg,int nField,LPSTR lpNet,LPINT lpError);

BOOL FAR PASCAL MfNumberToDouble(LPMFNUMBER lpNumber,double far *lpDouble);
BOOL FAR PASCAL MfDoubleToNumber(LPMFNUMBER lpNumber,double far *lpDouble);

#ifdef __cplusplus
}
#endif
```

```

/*****STRUCT.H FILE*****/
/*****/

// This file contains structure definitions for EAMATE data used
// in usercode.c and custom.c

// Revised by LLD on 10/19/93 to incorporate sequence number and new query parameters

// Revised by LLD on 11/30/93 to incorporate statistical structures

/** Below is a list of abbreviations that are used in the **/
/** naming of the fields. If an abbreviation is not used **/
/** the full term appears in the field name **/

/*****/
/** Add Address **/
/** Amt Amount **/
/** An Annuity **/
/** Ann Annual **/
/** Comp Compensation **/
/** Corr Correct **/
/** Def Deferred **/
/** Diff Difference **/
/** Dis District **/
/** EIN Employee Identification Number **/
/** Emplmt Employment **/
/** Empr Employer **/
/** Estab Established **/
/** Fed Federal **/
/** Grp Group **/
/** ID Identification **/
/** Ind Indicator **/
/** Ins Insurance **/
/** Int Internal **/
/** Liab Liability **/
/** Lib Library **/
/** Lim Limitation **/
/** Loc Local **/
/** MRN Microfilm Reference Number **/
/** Neg Negative **/
/** Num Number **/
/** Orig Original **/
/** Pen Pension **/
/** Proc Processed **/
/** Rec Record **/
/** Rep Reported **/
/** Ret Retirement **/
/** Rev Revenue **/
/** Sec Security **/
/** Soc Social **/
/** SSN Social Security Number **/
/** St State **/
/** Stat Statutory **/
/** Tps Tips **/
/** Trm Term **/
/** Unc Uncollected **/
/** Wgs Wages **/
/** Wheld Withheld **/
/*****/

#include <time.h>

/***** *****/
```

```
/** This define section was used when test data was built, **/
/** but is not being used for the current user interface **/
```

```
/** It may be used in the future to differentiate between **/
/** textual data and image data when employee detail info **/
/** is displayed **/
```

```
/** this will be the first byte in each type of record **/
```

```
#define W2EH 0          /** EAMATE W2 Empr Header Type **/
#define W2EI 1          /** EAMATE W2 Emp Info Type **/
#define W2IT 2          /** EAMATE W2 Intermed Total Type **/
#define W2FT 3          /** EAMATE W2 Final Total Type **/
#define W2CE 4          /** EAMATE W2 Cum EIN Type **/
#define W2CEH 5         /** EAMATE W2C Empr Header Type **/
#define W2CEI 6         /** EAMATE W2C Emp Info Type **/
#define W2CFT 7         /** EAMATE W2C Final Total Type **/
#define AWR_W2EH 8      /** AWR W2 Empr Header Type **/
#define AWR_W2EI 9      /** AWR W2 Emp Info Type **/
#define AWR_W2IT 10     /** AWR W2 Intermed Total Type **/
#define AWR_W2FT 11     /** AWR W2 Final Total Type **/
#define AWR_W2CE 12     /** AWR W2 Cumulative EIN **/
#define AWR_W2CEH 13    /** AWR W2 Empr Header Type **/
#define AWR_W2CEI 14    /** AWR W2 Emp Info Type **/
#define AWR_W2CFT 15    /** AWR W2 Final Total Type **/
```

```
/****** END OF NON-USED SECTION *****/
```

```
//NOTE: dummy characters are used to compensate for byte alignment on the SUN,
//       unsigned char of 4 is used in this file where long is used on the SUN,
//       the byte alignment for long on the SUN is every 4 bytes, therefore
//       the number of bytes previous to an unsigned char of 4 in this file
//       must be a multiple of 4 - i.e. before platter_side below, a total of
//       208 bytes (including one dummy byte) are present before the structure
//       member platter_side
```

```
/**NOTE: ALL DOLLAR VALUES INCLUDE THE DECIMAL**/
```

```
struct W2EmprInfo {          /** structure composed of header info, final totals, and
                             cum ein, if it exists; server will pass this entire struc
ture
```

```
                             and client will hold info, until a new ein is requested **
```

```
/
```

```
    /** empr header info **/
```

```
    char EIN[11];
```

```
    char EtabNumber[5];
```

```
    char ReportYear[5];
```

```
    char ProcessYear[5];
```

```
    char TapeLibNum[7];
```

```
    char TypeEmpr[2];
```

```
    char NameCode[2];
```

```
    char OtherEIN[10];
```

```
    char MRN[12];
```

```
    char EndMRN[12];
```

```
    char seq_no[4];
```

```
    char EmprName[48];
```

```
    char EmprStreetAdd[41];
```

```
    char EmprCity[26];
```

```
    char EmprState[11];
```

```
    char EmprZipCode [6];
```

```
    char dummy;
```

```
    /**compensation for SUN byte alignment
```



```
    unsigned char platter_side[4];
    unsigned char initials[4];
    unsigned char num_recs[4];
    unsigned char browse_start[4];

    /** final totals **/
    char ProcFICAWages[15];  /** all of these are dollar values **/
    char RepFICAWages[15];
    char ProcFICATips[14];
    char RepFICATips[14];
    char ProcWgsTpsOther[15];
    char RepWgsTpsOther[15];
    char ProcFedTaxWheld[14];
    char RepFedTaxWheld[14];
    char ProcFICATaxWheld[14];
    char RepFICATaxWheld[14];
    char ProcEarnInc[14];
    char RepEarnInc[14];
    char ProcItems[8];      /** this is not a dollar value **/
    char RepItems[8];      /** this is not a dollar value **/

    char ProcDefComp[15];
    char RepDefComp[15];
    char ProcNonequal[15];
    char RepNonequal[15];
    char ProcMedWages[15];
    char RepMedWages[15];
    char ProcMedTax[15];
    char RepMedTax[15];

    /** Cum EIN Totals **/
    char cflag[1];          /** ascii 0 or 1 for true or false **/
    char cProcFICAWages[15]; /** all of these are dollar values **/
    char cProcFICATips[14];
    char cProcWgsTpsOther[15];
    char cProcFedTaxWheld[14];
    char cProcFICATaxWheld[14];
    char cProcEarnInc[14];
    char cProcItems[8];    /** this is not a dollar value **/
};

struct W2EmprHeader {      /** ADJUSTED per 1991 data **/
    char EIN[11];
    char EstabNumber[5];
    char ReportYear[5];
    char ProcessYear[5];
    char TapeLibNum[7];
    char TypeEmpr[2];
    char NameCode[2];
    char OtherEIN[10];
    char MRN[12];
    char EndMRN[12];
    char seq_no[4];
    char EmprName[48];
    char EmprStreetAdd[41];
    char EmprCity[26];
    char EmprState[11];
    char EmprZipCode [6];
    char dummy;
    unsigned char platter_side[4];
    unsigned char num_recs[4];
    unsigned char final_offset[4];
    unsigned char cum_offset[4];
};
```

```
struct W2EmpInfo {          /** ADJUSTED per 1991 data **/
    char MRN[12];
    char EmpSSN[12];        /** two spaces included **/
    char EmpName[28];
    char PensionInd[2];
    char DefCompInd[2];
    char AnnFICAWages[9];    /** dollar value **/
    char AnnFICATips[9];     /** dollar value **/
    char AnnWgsTpsOther[11]; /** dollar value **/
    char FedTaxWheld[11];    /** dollar value **/
    char FICATaxWheld[8];    /** dollar value **/
    char AdvEarnInc[9];      /** dollar value **/
    char MedWages[10];       /** dollar value **/
    char MedTax[8];          /** dollar value **/
    char ControlNumber[8];

    char EmpStreetAdd[28];
    char DepCare[9];         /** dollar value **/
    char AllocTips[9];       /** dollar value **/
    char EmprGrpTrmInsCost[9]; /** dollar value **/
    char UncFICATipTax[9];   /** dollar value **/

    char EmpCity[19];
    char EmpState[3];
    char EmpZipCode[6];
    char DefCompAmt[11];     /** dollar value **/
    char StatEmpCode[2];
    char FringeBenefits[11]; /** dollar value **/
    char nqsec[11];          /** dollar value **/
    char nqnot[11];          /** dollar value **/
};

struct W2Browse {          /** added 11/16/92 - Employee Browse information t
    hat
                                will be displayed in the listbox **/

    char EmpSSN[12];
    char EmpName[28];
    char AnnFICAWages[9];    /** dollar value **/
    char AnnFICATips[9];     /** dollar value **/
    char FICATaxWheld[8];    /** dollar value **/
    char AnnWgsTpsOther[11]; /** dollar value **/
    char MRN[12];
    char seq_no[4];
    char wage_type[2];
    char dummy;              /** compensation for byte alignment on the SUN **/
    unsigned char record_loc[4]; /** offset for record location **/
};

// struct W2IntermedTotal deleted 10/19/93 - not used by the interface
// this structure is defined on the server side and used for printing
// an entire employer report which is also done on the server side

struct W2FinalTotal {      /** ADJUSTED for 1991 data **/
    char ProcFICAWages[15];  /** all of these are dollar values **/
    char RepFICAWages[15];
    char ProcFICATips[14];
    char RepFICATips[14];
    char ProcWgsTpsOther[15];
    char RepWgsTpsOther[15];
    char ProcFedTaxWheld[14];
    char RepFedTaxWheld[14];
    char ProcFICATaxWheld[14];
```

```
    char RepFICATaxWheld[14];
    char ProcEarnInc[14];
    char RepEarnInc[14];
    char ProcItems[8];          /** this is not a dollar value **/
    char RepItems[8];          /** this is not a dollar value **/

    char ProcDefComp[15];      /** all of these are dollar values **/
    char RepDefComp[15];
    char ProcNonequal[15];
    char RepNonequal[15];
    char ProcMedWages[15];
    char RepMedWages[15];
    char ProcMedTax[15];
    char RepMedTax[15];
};

struct W2CumEIN {              /** ADJUSTED for 1991 data **/
    char ProcFICAWages[15];    /** all of these are dollar values **/
    char ProcFICATips[14];
    char ProcWgsTpsOther[15];
    char ProcFedTaxWheld[14];
    char ProcFICATaxWheld[14];
    char ProcEarnInc[14];
    char ProcItems[8];        /** this is not a dollar value **/
};

struct query {                /** added 1/5/93 by LLD, query information
                                written to query*.txt which server
                                utilizes to conduct the search **/

                                /** updated 10/19/93 to add seq_no & offset **/

    char Year[5];
    char EIN[11];
    char Estab[5];
    char seq_no[4];           //sequence number of employer report
    char FName[13];
    char LName[16];
    char SSN[12];
    char offset[30];         //browse offset or MRN
};

struct in_stat {              //added 11/30/93, internal structure to
                                //app that will hold related statistics per
                                //each single query or browse query

    char interrupted;        //yes = 1, no = 0, were there any interruptions?
    char resolved;           //yes = 1, no = 0, was case resolved?
    char outlier;            //yes = 1, no = 0, is snet_time > 1/2 hour?

    //use difftime() function to calculate net times between two times
    double snet_time;         //stop - start_single
    double qnet_time;         //stop - start_qinfo
    double bnet_time;         //stop_browse - start_browse

    long add_match;           //count of how many times addt'l matches is pushed

    time_t start_single;      //when single query is chosen
    time_t start_qinfo;       //when qinfo is opened
    time_t stop_single;       //when close or potential blanket is selected
```



```
time_t start_browse; //when browse report is chosen
time_t stop_browse; //when close is chosen from brinfo
};

struct ex_stat { //added 11/30/93, structure that will be
//written to stat#.fil, this structure
//holds aggregate statistics

//TOTALS USED FOR CALCULATING AVERAGES AND PERCENTAGES

double br_tot_time; //running total of bnet_time

double rs_tot_time; //running total of snet_time of resolved cases
//without interruptions (if interrupted = 0)

double us_tot_time; //running total of snet_time of unresolved cases
//without interruptions (if interrupted = 0)

double rsi_tot_time; //running total of snet_time of resolved cases
//with interruptions (if interrupted = 1)

double usi_tot_time; //running total of snet_time of unresolved cases
//with interruptions (if interrupted = 1)

double rq_tot_time; //running total of qnet_time of resolved cases
//without interruptions (if interrupted = 0)

double uq_tot_time; //running total of qnet_time of unresolved cases
//without interruptions (if interrupted = 0)

double rqi_tot_time; //running total of qnet_time of resolved cases
//with interruptions (if interrupted = 1)

double uqi_tot_time; //running total of qnet_time of unresolved cases
//with interruptions (if interrupted = 1)

long rtot_non_interrupt; //running total of resolved cases
//without interruptions

long utot_non_interrupt; //running total of unresolved cases
//without interruptions

long tot_non_interrupt; //running total of number of cases
//without interruptions
// rtot_non_interrupt + utot_non_interrupt

long rtot_interrupt; //running total of resolved cases
//with interruptions

long utot_interrupt; //running total of unresolved cases
//with interruptions

long tot_interrupt; //running total of number of cases
//with interruptions
// rtot_interrupt + utot_interrupt

long tot_browse; //running total of number of times
//browse report is chosen and completed
//incremented if stop browse != 0
```

```
long tot_cases;           //running total of number of cases
                           // tot_resolved + tot_unresolved

long tot_non_cases;       //running total of number of non-cases,
                           //cases with a start time but no stop time

long tot_resolved;        //running total of resolved cases
                           // rtot_non_interrupt + rtot_interrupt

long tot_unresolved;      //running total of unresolved cases
                           // utot_non_interrupt + utot_interrupt

long tot_outliers;        //running total of number of cases
                           //identified as outliers
                           //cases with an snet_time > 1/2 hour

long tot_add_records;     //running total of number of times add
                           //records is pushed during browse report

long tot_add_matches;     //running total of number of times
                           //additional matches is pressed

long rtot_add_match;      //running total of number of times add matches
                           //is pressed for resolved cases

long rtot_add_match0;     //running total of number of resolved cases
                           //in which addt'l matches was not pushed

long rtot_add_match1;     //running total of number of resolved cases
                           //in which addt'l matches was pressed once

long rtot_add_match2;     //running total of number of resolved cases
                           //in which addt'l matches was pressed twice

long rtot_add_match3;     //running total of number of resolved cases in
                           //which addt'l matches was pressed 3 times

long rtot_add_match4_plus; //running total of number of resolved cases in
                           //which addt'l matches was pressed 4 or more times

long utot_add_match;      //running total of number of times add matches
                           //is pressed for unresolved cases

long utot_add_match0;     //running total of number of unresolved cases
                           //in which addt'l matches was not pushed

long utot_add_match1;     //running total of number of unresolved cases
                           //in which addt'l matches was pressed once

long utot_add_match2;     //running total of number of unresolved cases
                           //in which addt'l matches was pressed twice

long utot_add_match3;     //running total of number of unresolved cases in
                           //which addt'l matches was pressed 3 times

long utot_add_match4_plus; //running total of number of unresolved cases in
                           //which addt'l matches was pressed 4 or more times

//AVERAGES and PERCENTAGES

double avg_br_time;       // avg total browse report time
                           // br_tot_time / tot_browse
```

```
double avg_add_record;           // avg add_records per browse report
                                  // tot_add_records / tot_browse

double rs_avg_time;              // avg total single query time for resolved
                                  // cases without interruptions
                                  // rs_tot_time / rtot_non_interrupt

double rsi_avg_time;             // avg total single query time for resolved
                                  // cases with interruptions
                                  // rsi_tot_time / rtot_interrupt

double us_avg_time;              // avg total single query time for unresolved
                                  // cases without interruptions
                                  // us_tot_time / utot_non_interrupt

double usi_avg_time;             // avg total single query time for unresolved
                                  // cases with interruptions
                                  // usi_tot_time / utot_interrupt

double rq_avg_time;              // avg browse single query time for resolved
                                  // cases without interruptions
                                  // rq_tot_time / rtot_non_interrupt

double rqi_avg_time;             // avg browse single query time for resolved
                                  // cases with interruptions
                                  // rqi_tot_time / rtot_interrupt

double uq_avg_time;              // avg browse single query time for unresolved
                                  // cases without interruptions
                                  // uq_tot_time / utot_non_interrupt

double uqi_avg_time;            // avg browse single query time for unresolved
                                  // cases with interruptions
                                  // uqi_tot_time / utot_interrupt

double ravg_add_match;           // avg number of addt'l matches per resolved case
                                  // sum of all rtot_add_match* / tot_resolved

double uavg_add_match;           // avg number of addt'l matches per unresolved cas
                                  // sum of all utot_add_match* / tot_unresolved

double avg_add_match;            // avg number of addt'l matches per case
                                  // tot_add_matches / tot_cases

double rper_add_match0;          // percentage of resolved cases where
                                  // addt'l matches was not pressed
                                  // rtot_add_match0 / tot_resolved

double rper_add_match1;          // percentage of resolved cases where
                                  // addt'l matches was pressed once
                                  // rtot_add_match1 / tot_resolved

double rper_add_match2;          // percentage of resolved cases where
                                  // addt'l matches was pressed twice
                                  // rtot_add_match2 / tot_resolved

double rper_add_match3;          // percentage of resolved cases where
                                  // addt'l matches was pressed 3 times
                                  // rtot_add_match3 / tot_resolved
```



```
double rper_add_match4_plus;    // percentage of resolved cases where
                                // addt'l matches was pressed 4 or more times
                                // rtot_add_match4_plus / tot_resolved

double uper_add_match0;         // percentage of unresolved cases where
                                // addt'l matches was not pressed
                                // utot_add_match0 / tot_unresolved

double uper_add_match1;         // percentage of unresolved cases where
                                // addt'l matches was pressed once
                                // utot_add_match1 / tot_unresolved

double uper_add_match2;         // percentage of unresolved cases where
                                // addt'l matches was pressed twice
                                // utot_add_match2 / tot_unresolved

double uper_add_match3;         // percentage of unresolved cases where
                                // addt'l matches was pressed 3 times
                                // utot_add_match3 / tot_unresolved

double uper_add_match4_plus;    // percentage of unresolved cases where
                                // addt'l matches was pressed 4 or more times
                                // utot_add_match4_plus / tot_unresolved

//UI USAGE TOTALS PER SELECTED OPERATION

long tot_single_query;          //running total of single query selection

long tot_browse_report;         //running total of browse report selection

long tot_print_report;          //running total of print report selection

long tot_blanket;               //running total of potential blanket selection

long tot_diff_report;           //running total of select different report
                                //operation (arrow icon on qinfo)

long tot_qp;                    //running total of displaying current query
                                //parameters operation (QP icon on qinfo)

long tot_er_detail;             //running total of employer detail selection

long tot_ee_detail;             //running total of employee detail selection

long tot_final;                 //running total of report totals selection

long tot_pr_er_detail;          //running total of print employer
                                //detail selection

long tot_pr_ee_detail;          //running total of print employee
                                //detail selection

long tot_ee_det_printed;        //running total of actual number of
                                //employee details printed,
                                //this is different that just selecting
                                //the pushbutton operation because printing
                                //multiple copies of employee details is
                                //an option, therefore this total will be
                                //accumulated by adding up atoi(cnum)

long tot_pr_final;              //running total of print final totals selection
```

```
long tot_pr_blanket;           //running total of print blanket selection
long tot_pr_report;           //running total of OK selection in print
                               //report data entry dialog
```

} ;





ADDMSG DIALOG 126,62,120,79

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS

BEGIN

CONTROL "Continue",2,"Button",WS\_TABSTOP | WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,42,53,35,14

CONTROL "Additional Matches to be Implemented at a Later Date",100,"STATIC",SS\_CENTER | WS\_CHILD,28,9,64,34

CONTROL "WMPGRAPHIC",101,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,92,34,16,16

CONTROL "WMPGRAPHIC",102,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,11,34,16,16

END

STATISTICS DIALOG 108,62,190,69

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS | WS\_CLIPCHILDREN | WS\_CAPTION

CAPTION "Report Statistics"

BEGIN

CONTROL "OK",1,"Button",BS\_DEFPUSHBUTTON | WS\_TABSTOP | WS\_CHILD | WS\_VISIBLE,58,48,30,14

CONTROL "Cancel",2,"Button",WS\_TABSTOP | WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,100,48,30,14

CONTROL "TO BE IMPLEMENTED AT A LATER DATE",101,"static",SS\_CENTER | WS\_CHILD,53,13,81,20

CONTROL "WMPGRAPHIC",102,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,6,24,16,16

CONTROL "WMPGRAPHIC",104,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,151,8,16,16

CONTROL "WMPGRAPHIC",105,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,22,8,16,16

CONTROL "WMPGRAPHIC",106,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,166,24,16,16

END

PTOTALS DIALOG 40,19,112,63

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

CONTROL "Continue",101,"button",WS\_TABSTOP | WS\_CHILD | BS\_PUSHBUTTON,39,37,34,10

CONTROL "Report Totals Now Printing",102,"static",SS\_CENTER | WS\_CHILD,30,18,50,17

CONTROL "WMPGRAPHIC",103,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,0,0,16,16

CONTROL "WMPGRAPHIC",104,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,0,32,16,16

CONTROL "WMPGRAPHIC",105,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,0,16,16,16

CONTROL "WMPGRAPHIC",106,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,16,0,16,16

CONTROL "WMPGRAPHIC",107,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,32,0,16,16

CONTROL "WMPGRAPHIC",108,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,48,0,16,16

CONTROL "WMPGRAPHIC",109,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,80,0,16,16

CONTROL "WMPGRAPHIC",110,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,0,48,16,16

CONTROL "WMPGRAPHIC",111,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,64,0,16,16

CONTROL "WMPGRAPHIC",112,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,96,0,16,16

CONTROL "WMPGRAPHIC",113,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,96,16,16,16

CONTROL "WMPGRAPHIC",115,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,96,48,16,16

CONTROL "WMPGRAPHIC",116,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,16,48,16,16

CONTROL "WMPGRAPHIC",117,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,32,48,16,16

CONTROL "WMPGRAPHIC",118,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,48,48,16,16

CONTROL "WMPGRAPHIC",119,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,64,48,16,16

CONTROL "WMPGRAPHIC",120,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,80,48,16,16

CONTROL "WMPGRAPHIC",121,"button",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD | WS\_VISIBLE,96,32,16,16

END

NMSG DIALOG 164,25,120,79

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS

BEGIN

CONTROL "Continue",2,"Button",WS\_TABSTOP | WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,42,59,35,14

CONTROL "Your must enter both the first and last name of the employee.",100,"STATIC",SS\_CENTER | WS\_CHILD,28,12,64,34

CONTROL "WMPGRAPHIC",101,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,97,42,16,16

CONTROL "WMPGRAPHIC",102,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,7,42,16,16

END

QMSG DIALOG 164,25,120,79

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS

BEGIN

CONTROL "Continue",2,"Button",WS\_TABSTOP | WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,42,59,35,14

CONTROL "You must enter either the Employee Name or the Full SSN or both.",100,"STATIC",SS\_CENTER | WS\_CHILD,28,6,64,41

CONTROL "WMPGRAPHIC",101,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,93,42,16,16

CONTROL "WMPGRAPHIC",102,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,7,42,16,16

END

MAIN DIALOG 0,0,404,302

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS | WS\_CLIPCHILDREN

BEGIN

CONTROL "Single Query",125,"button",WS\_GROUP | WS\_TABSTOP | WS\_CHILD | BS\_PUSHBUTTON,0,0,80,24

CONTROL "Report Statistics",130,"button",WS\_TABSTOP | WS\_CHILD | BS\_PUSHBUTTON,80,0,80,24

CONTROL "Browse Report",126,"button",WS\_TABSTOP | WS\_CHILD | BS\_PUSHBUTTON,160,0,80,24

CONTROL "Print Report",127,"button",WS\_TABSTOP | WS\_CHILD | BS\_PUSHBUTTON,240,0,80,24

CONTROL "Exit Program",128,"button",WS\_TABSTOP | WS\_CHILD | BS\_PUSHBUTTON,320,0,80,24

END

HDETAIL DIALOG 32,33,237,112

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS | WS\_CLIPCHILDREN | WS\_CAPTION

CAPTION "Employer Header Detail"

BEGIN

CONTROL "Continue",101,"BUTTON",WS\_TABSTOP | WS\_CHILD | BS\_PUSHBUTTON,73,95,40,16

CONTROL "Print",IDPRINTHD,"BUTTON",WS\_TABSTOP | WS\_CHILD | BS\_PUSHBUTTON,130,95,40,16

CONTROL "",ID\_HDetail,"STATIC",WS\_CHILD | SS\_LEFT,4,4,229,87

CONTROL "WMPGRAPHIC",104,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,29,95,16,16

CONTROL "WMPGRAPHIC",106,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,196,95,16,16

END

NMount DIALOG 48,20,94,105

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

CONTROL "OK",1,"Button",BS\_DEFPUSHBUTTON | WS\_TABSTOP | WS\_CHILD | WS\_VISIBLE,31,80,30,14

CONTROL "Optical Disks Are Not Mounted",102,"static",SS\_CENTER | WS\_CHILD,19,32,56,17

CONTROL "WMPGRAPHIC",103,"BUTTON",BS\_OWNERDRAW | WS\_GROUP | WS\_TABSTOP | WS\_CHILD,18,5,16,16

CONTROL "WMPGRAPHIC",105,"BUTTON",BS\_OWNERDRAW | WS\_GROUP | WS\_TABSTOP | WS\_CHILD,61,5,16,16

CONTROL "See System Administrator",106,"static",SS\_CENTER | WS\_CHILD,22,53,52,18

END

QUERYERROR DIALOG 164,25,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

CONTROL "OK",1,"Button",BS\_DEFPUSHBUTTON | WS\_TABSTOP | WS\_CHILD | WS\_VISIBLE,37,99,30,14

CONTROL "A System Error Has Occurred",102,"static",SS\_CENTER | WS\_CHILD,16,38,76,19

CONTROL "WMPGRAPHIC",103,"BUTTON",BS\_OWNERDRAW | WS\_GROUP | WS\_TABSTOP | WS\_CHILD,24,1,0,16,16

CONTROL "See System Administrator",104,"static",SS\_CENTER | WS\_CHILD,15,66,77,24

CONTROL "WMPGRAPHIC",105,"BUTTON",BS\_OWNERDRAW | WS\_GROUP | WS\_TABSTOP | WS\_CHILD,67,1,0,16,16

END

BLANKERR DIALOG 141,20,94,105

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

CONTROL "OK",1,"Button",BS\_DEFPUSHBUTTON | WS\_TABSTOP | WS\_CHILD | WS\_VISIBLE,31,80,30



```
,14
CONTROL "Blanket File Does Not Exist",102,"static",SS_CENTER | WS_CHILD,16,29,62,19
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,18,5
,16,16
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,61,5
,16,16
CONTROL "See System Administrator",106,"static",SS_CENTER | WS_CHILD,16,51,62,19
END
```

EDETAIL DIALOG 24,22,255,186

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS | WS\_CLIPCHILDREN | WS\_CAPTION  
CAPTION "Employee Detail"

BEGIN

```
CONTROL "Continue",101,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,73,168,40,16
CONTROL "Print",IDPRINTED,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,144,168,40,16
CONTROL "",ID_EDetail,"STATIC",WS_CHILD | SS_LEFT,4,4,247,161
CONTROL "WMPGRAPHIC",104,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,34,168,16,16
CONTROL "WMPGRAPHIC",106,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,212,168,16,16
END
```

PEDETAIL DIALOG 100,43,112,63

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

```
CONTROL "Continue",IDCONT,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,39,37,34,10
CONTROL "Employee Detail Now Printing",102,"static",SS_CENTER | WS_CHILD,28,18,57,17
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,0,0,16,16
CONTROL "WMPGRAPHIC",104,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,0,32,16,16
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,0,16,16,16
CONTROL "WMPGRAPHIC",106,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,16,0,16,16
CONTROL "WMPGRAPHIC",107,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,32,0,16,16
CONTROL "WMPGRAPHIC",108,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,48,0,16,16
CONTROL "WMPGRAPHIC",109,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,80,0,16,16
CONTROL "WMPGRAPHIC",110,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,0,48,16,16
CONTROL "WMPGRAPHIC",111,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,64,0,16,16
CONTROL "WMPGRAPHIC",112,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,96,0,16,16
CONTROL "WMPGRAPHIC",113,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,96,16,16,16
CONTROL "WMPGRAPHIC",114,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,96,32,16,16
CONTROL "WMPGRAPHIC",115,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,96,48,16,16
CONTROL "WMPGRAPHIC",116,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,16,48,16,16
CONTROL "WMPGRAPHIC",117,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,32,48,16,16
CONTROL "WMPGRAPHIC",118,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,48,48,16,16
CONTROL "WMPGRAPHIC",119,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,64,48,16,16
CONTROL "WMPGRAPHIC",120,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,80,48,16,16
END
```

PHDETAIL DIALOG 92,25,112,63

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

```
CONTROL "Continue",IDCONT,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,39,37,34,10
CONTROL "Header Detail Now Printing",102,"static",SS_CENTER | WS_CHILD,30,18,50,17
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,0,0,16,16
CONTROL "WMPGRAPHIC",104,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,0,32,16,16
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,0,16,16,16
CONTROL "WMPGRAPHIC",106,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,16,0,16,16
CONTROL "WMPGRAPHIC",107,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,32,0,16,16
CONTROL "WMPGRAPHIC",108,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,48,0,16,16
CONTROL "WMPGRAPHIC",109,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,80,0,16,16
CONTROL "WMPGRAPHIC",110,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,0,48,16,16
CONTROL "WMPGRAPHIC",111,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,64,0,16,16
CONTROL "WMPGRAPHIC",112,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,96,0,16,16
CONTROL "WMPGRAPHIC",113,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,96,16,16,16
CONTROL "WMPGRAPHIC",114,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,96,32,16,16
CONTROL "WMPGRAPHIC",115,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,96,48,16,16
CONTROL "WMPGRAPHIC",116,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,16,48,16,16
```



```

CONTROL "WMPGRAPHIC",117,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,32,48,16,16
CONTROL "WMPGRAPHIC",118,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,48,48,16,16
CONTROL "WMPGRAPHIC",119,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,64,48,16,16
CONTROL "WMPGRAPHIC",120,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,80,48,16,16

```

END

PBLANKET DIALOG 100,46,128,80

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGMFRAME

BEGIN

```

CONTROL "Continue",IDCONT,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,47,47,34,12
CONTROL "Blanket Report Now Printing",102,"static",SS_CENTER | WS_CHILD,38,23,53,17
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,0,0,16,16
CONTROL "WMPGRAPHIC",104,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,0,32,16,16
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,0,16,16,16
CONTROL "WMPGRAPHIC",106,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,16,0,16,16
CONTROL "WMPGRAPHIC",107,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,32,0,16,16
CONTROL "WMPGRAPHIC",108,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,48,0,16,16
CONTROL "WMPGRAPHIC",109,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,80,0,16,16
CONTROL "WMPGRAPHIC",110,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,0,64,16,16
CONTROL "WMPGRAPHIC",111,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,64,0,16,16
CONTROL "WMPGRAPHIC",112,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,96,0,16,16
CONTROL "WMPGRAPHIC",113,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,112,16,16,16
CONTROL "WMPGRAPHIC",114,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,112,32,16,16
CONTROL "WMPGRAPHIC",115,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,112,0,16,16
CONTROL "WMPGRAPHIC",116,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,16,64,16,16
CONTROL "WMPGRAPHIC",117,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,32,64,16,16
CONTROL "WMPGRAPHIC",118,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,48,64,16,16
CONTROL "WMPGRAPHIC",119,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,64,64,16,16
CONTROL "WMPGRAPHIC",120,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,112,64,16,16
CONTROL "WMPGRAPHIC",121,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,112,48,16,16
CONTROL "WMPGRAPHIC",122,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,80,64,16,16
CONTROL "WMPGRAPHIC",123,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,0,48,16,16
CONTROL "WMPGRAPHIC",124,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,96,64,16,16
CONTROL "WMPGRAPHIC",125,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,16,16,16,16
CONTROL "WMPGRAPHIC",126,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,16,32,16,16
CONTROL "WMPGRAPHIC",127,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,16,48,16,16
CONTROL "WMPGRAPHIC",128,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,96,16,16,16
CONTROL "WMPGRAPHIC",129,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,96,32,16,16
CONTROL "WMPGRAPHIC",130,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,96,48,16,16

```

END

TOTALS DIALOG 32,6,269,239

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS | WS\_CLIPCHILDREN | WS\_CAPTION

CAPTION "Final Totals"

BEGIN

```

CONTROL "Continue",ID_CONT,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,64,220,40,16
CONTROL "Print",IDPRINTTOT,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,165,220,40,1

```

6

```

CONTROL "",ID_Tot,"STATIC",WS_CHILD | SS_LEFT,8,38,254,177
CONTROL "",ID_TotEIN,"STATIC",WS_CHILD | SS_LEFT,6,3,87,9
CONTROL "",ID_TotRpt,"STATIC",WS_CHILD | SS_LEFT,216,3,48,9
CONTROL "TYPE:",101,"STATIC",WS_CHILD | SS_LEFT,22,23,21,9
CONTROL "PROCESSED:",107,"STATIC",WS_CHILD | SS_LEFT,110,23,44,9
CONTROL "REPORTED:",108,"STATIC",WS_CHILD | SS_LEFT,208,23,40,9
CONTROL "",109,"STATIC",WS_CHILD | SS_LEFT,8,18,254,2
CONTROL "",110,"STATIC",WS_CHILD | SS_LEFT,8,33,254,2
CONTROL "",111,"STATIC",WS_CHILD | SS_LEFT,7,215,257,2
CONTROL "",112,"STATIC",WS_CHILD | SS_LEFT,6,18,2,199
CONTROL "",113,"STATIC",WS_CHILD | SS_LEFT,262,18,2,199

```

END

YEAR\_ERR DIALOG 164,25,132,87

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGMFRAME

BEGIN

```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,49,64,35,18
CONTROL "No Report Exists for that Year. Please Backspace and Re-Enter the Year.",100
,"STATIC",SS_CENTER | WS_CHILD,14,33,103,25
CONTROL "WMPGRAPHIC",101,"button",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,29,9,18,17
CONTROL "WAIT!",102,"STATIC",SS_CENTER | WS_CHILD,62,13,30,9
END
```

BLANKETERERROR DIALOG 119,15,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFRAME

BEGIN

```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,39,98,30,14
CONTROL "A System Error Occurred",102,"static",SS_CENTER | WS_CHILD,20,39,67,18
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,24,1
0,16,16
CONTROL "See System Administrator",104,"static",SS_CENTER | WS_CHILD,20,65,67,20
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,64,1
0,16,16
END
```

DATAERR DIALOG 164,25,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFRAME

BEGIN

```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,37,99,30,14
CONTROL "System Unable to Open Data File",102,"static",SS_CENTER | WS_CHILD,15,30,77,1
6
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,24,1
0,16,16
CONTROL "See System Administrator",104,"static",SS_CENTER | WS_CHILD,15,75,77,19
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,67,1
0,16,16
CONTROL "No Entries Were Added to the List Box",106,"static",SS_CENTER | WS_CHILD,15,5
2,77,16
END
```

ERRORFILE DIALOG 164,25,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFRAME

BEGIN

```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,37,99,30,14
CONTROL "System Unable to Open Local Error File",102,"static",SS_CENTER | WS_CHILD,8,3
9,92,19
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,24,1
0,16,16
CONTROL "See System Administrator",104,"static",SS_CENTER | WS_CHILD,15,67,77,19
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,67,1
0,16,16
END
```

NOMATCH DIALOG 164,25,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFRAME

BEGIN

```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,37,99,30,14
CONTROL "No Matches Found for this Query",102,"static",SS_CENTER | WS_CHILD,23,38,63,1
9
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,24,1
0,16,16
CONTROL "Please Check Parameters and Re-Enter",104,"static",SS_CENTER | WS_CHILD,15,63
,77,24
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,67,1
0,16,16
END
```

NOMORE DIALOG 102,15,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFRAME

BEGIN



```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,37,99,30,14
CONTROL "No More Matches Found",102,"static",SS_CENTER | WS_CHILD,15,41,79,11
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,29,1
0,16,16
CONTROL "No Entries Were Added to the List Box",106,"static",SS_CENTER | WS_CHILD,15,6
3,77,16
CONTROL "NOTE",107,"STATIC",SS_CENTER | WS_CHILD,51,13,27,11
END
```

QUERYTXT DIALOG 164,25,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,37,99,30,14
CONTROL "System Unable to Open Query*.Txt File",102,"static",SS_CENTER | WS_CHILD,8,39
,92,19
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,24,1
0,16,16
CONTROL "See System Administrator",104,"static",SS_CENTER | WS_CHILD,15,67,77,19
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,67,1
0,16,16
END
```

SYSERR DIALOG 132,20,94,105

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,31,80,30,14
CONTROL "A System Error Has Occurred",102,"static",SS_CENTER | WS_CHILD,16,29,62,19
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,18,5
,16,16
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,61,5
,16,16
CONTROL "See System Administrator",106,"static",SS_CENTER | WS_CHILD,16,51,62,19
END
```

QUERY DIALOG 81,48,226,118

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS | WS\_CAPTION

CAPTION "Please Enter Query Information"

BEGIN

```
CONTROL "M0100X;1,1400,410,0|QYear|RepYr Help|Gray Down||",IDMF_QYear,"MFEDIT",ES_LEFT
| WS_TABSTOP | WS_CHILD | WS_BORDER,54,5,101,13
CONTROL "M0100X;1,1418,E1A,0|QEIN|EINHelp|Gray Down||",IDMF_QEIN,"MFEDIT",ES_LEFT | WS
_TABSTOP | WS_CHILD | WS_BORDER,54,24,101,13
CONTROL "M0100X;1,400,D1A,0|QName|NameHelp|Gray Down||",IDMF_LName,"MFEDIT",ES_LEFT |
ES_UPPERCASE | ES_AUTOHSCROLL | WS_TABSTOP | WS_CHILD | WS_BORDER,54,62,101,13
CONTROL "M0100X;1,400,D1A,0|QName|NameHelp|Gray Down||",IDMF_FName,"MFEDIT",ES_LEFT |
ES_UPPERCASE | ES_AUTOHSCROLL | WS_TABSTOP | WS_CHILD | WS_BORDER,54,81,101,13
CONTROL "M0100X;1,418,E1E,0|QSSN|SSNHelp|Gray Down||",IDMF_QSSN,"MFEDIT",ES_LEFT | WS
_TABSTOP | WS_CHILD | WS_BORDER,54,100,101,13
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,174,29,39,19
CONTROL "Cancel",2,"Button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,174,68,39,19
CONTROL "M0100X;1,1408,E1A,0|QESTAB||Gray Down||",IDMF_QEstab,"MFEDIT",ES_LEFT | ES_AU
TOHSCROLL | WS_TABSTOP | WS_CHILD | WS_BORDER,54,43,101,13
CONTROL "YEAR",109,"STATIC",WS_CHILD | SS_LEFT,7,7,33,8
CONTROL "EIN",115,"STATIC",WS_CHILD | SS_LEFT,7,27,33,8
CONTROL "ESTAB#",119,"STATIC",WS_CHILD | SS_LEFT,7,47,33,8
CONTROL "LAST NAME",116,"STATIC",WS_CHILD | SS_LEFT,7,65,41,8
CONTROL "FIRST NAME",121,"STATIC",WS_CHILD | SS_LEFT,7,84,41,8
CONTROL "SSN",117,"STATIC",WS_CHILD | SS_LEFT,7,104,33,8
END
```

NCOPY DIALOG 52,20,99,87

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

```
CONTROL "M0100X;1,40A,C1A,0|NumCopies||Gray Down||",IDMF_NumCopies,"MFEDIT",ES_LEFT |
```



WS\_TABSTOP | WS\_CHILD | WS\_BORDER,41,40,17,12

CONTROL "OK",1,"Button",BS\_DEFPUSHBUTTON | WS\_TABSTOP | WS\_CHILD,35,64,30,14

CONTROL "Please Enter the Desired Number of Copies. Default is one.",103,"static",SS\_CENTER | WS\_CHILD,9,7,80,25  
END

REPORT DIALOG 92,53,213,84

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS | WS\_CAPTION

CAPTION "Please Enter Report Statistics Information"

BEGIN

CONTROL "M0100X;1,1418,F1A,0|QYear|RepYr Help|Gray Down||",IDMF\_RYear,"MFEDIT",ES\_LEFT  
| WS\_TABSTOP | WS\_CHILD | WS\_BORDER,58,11,101,14

CONTROL "M0100X;1,1418,F1A,0|QEIN|EINHelp|Gray Down||",IDMF\_REIN,"MFEDIT",ES\_LEFT | WS\_TABSTOP | WS\_CHILD | WS\_BORDER,58,33,101,14

CONTROL "M0100X;1,1408,E1A,0|QESTAB||Gray Down||",IDMF\_REstab,"MFEDIT",ES\_LEFT | ES\_AUTOHSCROLL | WS\_TABSTOP | WS\_CHILD | WS\_BORDER,58,55,101,14

CONTROL "OK",1,"Button",BS\_DEFPUSHBUTTON | WS\_TABSTOP | WS\_CHILD,169,15,37,17

CONTROL "Cancel",2,"Button",WS\_TABSTOP | WS\_CHILD | BS\_PUSHBUTTON,169,45,37,17

CONTROL "YEAR",109,"STATIC",WS\_CHILD | SS\_LEFT,7,15,33,8

CONTROL "EIN",115,"STATIC",WS\_CHILD | SS\_LEFT,7,37,33,8

CONTROL "ESTAB#",118,"STATIC",WS\_CHILD | SS\_LEFT,7,58,33,8

END

PINFO DIALOG 44,25,112,63

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

CONTROL "Continue",101,"button",WS\_TABSTOP | WS\_CHILD | BS\_PUSHBUTTON,39,33,34,10

CONTROL "Report Now Printing",102,"static",SS\_CENTER | WS\_CHILD,19,20,73,11

CONTROL "WMPGRAPHIC",103,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,0,0,16,16

CONTROL "WMPGRAPHIC",104,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,0,32,16,16

CONTROL "WMPGRAPHIC",105,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,0,16,16,16

CONTROL "WMPGRAPHIC",106,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,16,0,16,16

CONTROL "WMPGRAPHIC",107,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,32,0,16,16

CONTROL "WMPGRAPHIC",108,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,48,0,16,16

CONTROL "WMPGRAPHIC",109,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,80,0,16,16

CONTROL "WMPGRAPHIC",110,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,0,47,16,16

CONTROL "WMPGRAPHIC",111,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,64,0,16,16

CONTROL "WMPGRAPHIC",112,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,96,0,16,16

CONTROL "WMPGRAPHIC",113,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,96,16,16,16

CONTROL "WMPGRAPHIC",114,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,96,32,16,16

CONTROL "WMPGRAPHIC",115,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,96,48,16,16

CONTROL "WMPGRAPHIC",116,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,16,47,16,16

CONTROL "WMPGRAPHIC",117,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,32,47,16,16

CONTROL "WMPGRAPHIC",118,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,48,47,16,16

CONTROL "WMPGRAPHIC",119,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,64,47,16,16

CONTROL "WMPGRAPHIC",120,"BUTTON",BS\_OWNERDRAW | WS\_TABSTOP | WS\_CHILD,80,47,16,16

END

EINERROR DIALOG 164,25,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

CONTROL "OK",1,"Button",BS\_DEFPUSHBUTTON | WS\_TABSTOP | WS\_CHILD,41,99,30,14

CONTROL "No Report Exists for the Requested EIN and YEAR",102,"static",SS\_CENTER | WS\_CHILD,21,35,67,24

CONTROL "WMPGRAPHIC",103,"BUTTON",BS\_OWNERDRAW | WS\_GROUP | WS\_TABSTOP | WS\_CHILD,24,10,16,16

CONTROL "Press OK, Hit Return or ESC, and Re-Enter the EIN",104,"static",SS\_CENTER | WS\_CHILD,23,67,66,25

CONTROL "WAIT!",106,"STATIC",WS\_CHILD | SS\_LEFT,54,14,23,9

END

MRNERR DIALOG 164,25,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,41,99,30,14
CONTROL "No Report Exists for the Requested MRN",102,"static",SS_CENTER | WS_CHILD,27,
35,54,24
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,24,1
0,16,16
CONTROL "Press OK, Hit Return or ESC, and Re-Enter the MRN",104,"static",SS_CENTER | W
S_CHILD,22,67,66,25
CONTROL "WAIT!",106,"STATIC",WS_CHILD | SS_LEFT,54,14,27,10
END
```

PWERROR DIALOG 164,25,102,104

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,39,82,30,14
CONTROL "System Does Not Recognize This Password",102,"static",SS_CENTER | WS_CHILD,7,
31,88,18
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,21,7
,16,16
CONTROL "Please Re-Enter the Password",104,"static",SS_CENTER | WS_CHILD,15,57,77,18
CONTROL "WAIT!",106,"STATIC",WS_CHILD | SS_LEFT,54,12,25,10
END
```

QPARAM DIALOG 100,41,166,170

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS | WS\_CLIPCHILDREN

BEGIN

```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,72,149,39,14
CONTROL "YEAR",109,"STATIC",WS_CHILD | SS_LEFT,7,31,33,10
CONTROL "EIN",115,"STATIC",WS_CHILD | SS_LEFT,7,51,33,10
CONTROL "ESTAB#",119,"STATIC",WS_CHILD | SS_LEFT,7,71,33,10
CONTROL "LAST NAME",116,"STATIC",WS_CHILD | SS_LEFT,7,89,47,10
CONTROL "FIRST NAME",121,"STATIC",WS_CHILD | SS_LEFT,7,108,49,10
CONTROL "SSN",117,"STATIC",WS_CHILD | SS_LEFT,7,128,33,10
CONTROL "Single Query Parameters",124,"STATIC",WS_CHILD | SS_LEFT,31,7,116,11
CONTROL "",qp_year,"STATIC",WS_CHILD | SS_LEFT,70,30,89,11
CONTROL "",qp_ein,"STATIC",WS_CHILD | SS_LEFT,70,50,89,11
CONTROL "",qp_estab,"STATIC",WS_CHILD | SS_LEFT,70,70,89,11
CONTROL "",qp_lname,"STATIC",WS_CHILD | SS_LEFT,70,88,89,11
CONTROL "",qp_fname,"STATIC",WS_CHILD | SS_LEFT,70,107,89,11
CONTROL "",qp_ssn,"STATIC",WS_CHILD | SS_LEFT,70,127,89,11
END
```

SEQERR DIALOG 164,25,107,128

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,41,106,30,14
CONTROL "No Report Exists for the Requested Report Number",102,"static",SS_CENTER | WS
_CHILD,22,33,64,26
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,26,1
0,16,16
CONTROL "Clear This Message by Pressing OK and Re-Try Desired Operation with a Differe
nt Report Number",104,"static",SS_CENTER | WS_CHILD,13,66,83,35
CONTROL "WAIT!",106,"STATIC",WS_CHILD | SS_LEFT,56,14,27,10
END
```

SEQ DIALOG 146,13,99,106

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFAME

BEGIN

```
CONTROL "M0100X;0,40A,C1A,0|Custom5||Gray Down||",ID_SeqNo,"MFEDIT",ES_LEFT | ES_UPPER
CASE | WS_TABSTOP | WS_CHILD | WS_BORDER,33,51,35,13
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,35,82,30,14
CONTROL "Please Enter the Desired Employer Report Number. Default is the Current Repo
rt Number.",103,"static",SS_CENTER | WS_CHILD,5,7,89,36
END
```



PRINT DIALOG 92,45,213,123

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS | WS\_CAPTION

CAPTION "Please Enter Print Information"

BEGIN

CONTROL "M0100X;1,1418,E1A,0|Password|PWHelp|Gray Down||",IDMF\_Password,"MFEDIT",ES\_LEFT | ES\_PASSWORD | WS\_TABSTOP | WS\_CHILD | WS\_BORDER,58,9,101,14

CONTROL "M0100X;1,1400,410,0|QYear|RepYr Help|Gray Down||",IDMF\_PYear,"MFEDIT",ES\_LEFT | WS\_TABSTOP | WS\_CHILD | WS\_BORDER,58,31,101,14

CONTROL "M0100X;1,1418,F1A,0|QEIN|EINHelp|Gray Down||",IDMF\_PEIN,"MFEDIT",ES\_LEFT | WS\_TABSTOP | WS\_CHILD | WS\_BORDER,58,53,101,14

CONTROL "M0100X;0,1408,F1B,0|Sequence||Gray Down||",IDMF\_PSeq,"MFEDIT",ES\_LEFT | ES\_UPPERCASE | WS\_TABSTOP | WS\_CHILD | WS\_BORDER,58,97,101,14

CONTROL "OK",1,"Button",BS\_DEFPUSHBUTTON | WS\_TABSTOP | WS\_CHILD,170,28,37,17

CONTROL "Cancel",2,"Button",WS\_TABSTOP | WS\_CHILD | BS\_PUSHBUTTON,170,66,37,17

CONTROL "M0100X;1,40A,E1A,0|QESTAB||Gray Down||",IDMF\_PEstab,"MFEDIT",ES\_LEFT | WS\_CHILD | WS\_BORDER,58,75,101,14

CONTROL "PASSWORD",116,"STATIC",WS\_CHILD | SS\_LEFT,7,13,40,8

CONTROL "YEAR",109,"STATIC",WS\_CHILD | SS\_LEFT,7,35,33,8

CONTROL "EIN",115,"STATIC",WS\_CHILD | SS\_LEFT,7,57,33,8

CONTROL "ESTAB#",118,"STATIC",WS\_CHILD | SS\_LEFT,7,78,33,8

CONTROL "REPORT#",121,"STATIC",WS\_CHILD | SS\_LEFT,7,101,42,8

END

NULLPTR DIALOG 164,25,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGMFRAME

BEGIN

CONTROL "OK",1,"Button",BS\_DEFPUSHBUTTON | WS\_TABSTOP | WS\_CHILD,40,99,30,14

CONTROL "System Unable to Initialize Next Window Due to File Problems",102,"static",SS\_CENTER | WS\_CHILD,15,38,77,24

CONTROL "WMPGRAPHIC",103,"BUTTON",BS\_OWNERDRAW | WS\_GROUP | WS\_TABSTOP | WS\_CHILD,24,10,16,16

CONTROL "See System Administrator",104,"static",SS\_CENTER | WS\_CHILD,15,70,77,19

CONTROL "WMPGRAPHIC",105,"BUTTON",BS\_OWNERDRAW | WS\_GROUP | WS\_TABSTOP | WS\_CHILD,67,10,16,16

END

MISSINGFILE DIALOG 108,15,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGMFRAME

BEGIN

CONTROL "OK",1,"Button",BS\_DEFPUSHBUTTON | WS\_TABSTOP | WS\_CHILD,42,99,30,14

CONTROL "System Unable to Locate the Files Associated with the Next Window",102,"static",SS\_CENTER | WS\_CHILD,10,39,91,23

CONTROL "WMPGRAPHIC",103,"BUTTON",BS\_OWNERDRAW | WS\_GROUP | WS\_TABSTOP | WS\_CHILD,24,10,16,16

CONTROL "See System Administrator",104,"static",SS\_CENTER | WS\_CHILD,26,71,60,19

CONTROL "WMPGRAPHIC",105,"BUTTON",BS\_OWNERDRAW | WS\_GROUP | WS\_TABSTOP | WS\_CHILD,67,10,16,16

END

NEWBROWSE DIALOG 80,45,213,103

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS | WS\_CAPTION

CAPTION "Please Enter Browse Report Information"

BEGIN

CONTROL "M0100X;1,1418,F1A,0|QYear|RepYr Help|Gray Down||",IDMF\_BrYear,"MFEDIT",ES\_LEFT | WS\_TABSTOP | WS\_CHILD | WS\_BORDER,58,11,101,14

CONTROL "M0100X;1,1418,F1A,0|QEIN|EINHelp|Gray Down||",IDMF\_BrEIN,"MFEDIT",ES\_LEFT | WS\_TABSTOP | WS\_CHILD | WS\_BORDER,58,33,101,14

CONTROL "M0100X;1,1408,61B,0|StartRecord||Gray Down||",IDMF\_BrStart,"MFEDIT",ES\_LEFT | WS\_TABSTOP | WS\_CHILD | WS\_BORDER,58,77,101,14

CONTROL "OK",1,"Button",BS\_DEFPUSHBUTTON | WS\_TABSTOP | WS\_CHILD,169,21,37,17

CONTROL "Cancel",2,"Button",WS\_TABSTOP | WS\_CHILD | BS\_PUSHBUTTON,169,57,37,17

CONTROL "M0100X;1,408,E1A,0|QESTAB||Gray Down||",IDMF\_BrEstab,"MFEDIT",ES\_LEFT | ES\_AUTOHSCROLL | WS\_TABSTOP | WS\_CHILD | WS\_BORDER,58,55,101,14

CONTROL "YEAR",109,"STATIC",WS\_CHILD | SS\_LEFT,7,15,33,8



```
CONTROL "EIN",115,"STATIC",WS_CHILD | SS_LEFT,7,37,33,8
CONTROL "ESTAB#",118,"STATIC",WS_CHILD | SS_LEFT,7,58,33,8
CONTROL "BEGINNING MRN",121,"static",WS_CHILD | SS_LEFT,7,76,40,15
END

MAIN1 DIALOG 60,81,285,147
STYLE DS_MODALFRAME | WS_POPUP | WS_CLIPSIBLINGS
BEGIN
CONTROL "Continue",ID_CONT,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,136,124,37,1
4
CONTROL "WMPGRAPHIC",115,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,44,50,62,80
CONTROL "TO THE EAMATE PROTOTYPE",116,"static",WS_CHILD | SS_LEFT,26,38,105,10
CONTROL "1992",117,"static",SS_CENTER | WS_CHILD,206,112,35,9
CONTROL "Laura L. Downey, Computer Scientist",121,"STATIC",WS_CHILD | SS_LEFT,183,69,6
6,15
CONTROL "Natalie E. Willman,",123,"STATIC",WS_CHILD | SS_LEFT,183,46,72,9
CONTROL "Senior Computer Scientist",124,"STATIC",WS_CHILD | SS_LEFT,183,54,93,9
CONTROL "WMPGRAPHIC",100,"button",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,8,9,64,29
CONTROL "DESIGNED AND DEVELOPED BY",106,"static",SS_CENTER | WS_CHILD,161,30,122,13
CONTROL "National Institute of Standards and Technology",114,"static",SS_CENTER | WS_C
HILD,169,92,106,17
CONTROL "",132,"BUTTON",BS_GROUPBOX | WS_TABSTOP | WS_CHILD,134,118,41,22
END

USERERR DIALOG 82,0,107,148
STYLE WS_POPUP | WS_CLIPSIBLINGS | WS_DLGFRAME
BEGIN
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,39,125,30,14
CONTROL "System Unable to Assign User Number",102,"static",SS_CENTER | WS_CHILD,20,35,
67,18
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,24,1
0,16,16
CONTROL "See System Administrator",104,"static",SS_CENTER | WS_CHILD,20,58,67,18
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,64,1
0,16,16
CONTROL "Application Will Not Respond Properly Without Correct User Number",106,"stati
c",SS_CENTER | WS_CHILD,20,82,67,32
END

QUESTION DIALOG 40,0,225,261
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CLIPSIBLINGS
BEGIN
CONTROL "OK",1,"button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,174,31,41,18
CONTROL "",103,"BUTTON",BS_GROUPBOX | WS_TABSTOP | WS_CHILD,8,16,158,40
CONTROL "Did You Find the Person You Were Looking For?",105,"STATIC",WS_CHILD | SS_LEF
T,17,30,96,17
CONTROL "Yes",ID_Q1Yes,"BUTTON",BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP | WS_CHILD,
131,26,24,10
CONTROL "No",ID_Q1No,"BUTTON",BS_AUTORADIOBUTTON | WS_TABSTOP | WS_CHILD,131,41,24,10
CONTROL "",104,"BUTTON",BS_GROUPBOX | WS_TABSTOP | WS_CHILD,8,58,158,40
CONTROL "Were You Interrupted at Any Time During Your Search?",106,"STATIC",WS_CHILD |
SS_LEFT,17,72,96,17
CONTROL "Yes",ID_Q2Yes,"BUTTON",BS_AUTORADIOBUTTON | WS_GROUP | WS_TABSTOP | WS_CHILD,
131,68,24,10
CONTROL "No",ID_Q2No,"BUTTON",BS_AUTORADIOBUTTON | WS_TABSTOP | WS_CHILD,131,83,24,10
CONTROL "",114,"BUTTON",BS_GROUPBOX | WS_GROUP | WS_TABSTOP | WS_CHILD,8,98,158,30
CONTROL "Choose Yes or No to Each Question by Moving the Mouse Pointer onto the Circle
Located Next to Desired Choice and Pressing the Left Mouse Button Once",113,"STATIC",SS_C
ENTER | WS_CHILD,11,104,150,23
CONTROL "",115,"BUTTON",BS_GROUPBOX | WS_TABSTOP | WS_CHILD,172,53,45,55
CONTROL "Press OK When You Are Finished Answering the Questions",111,"STATIC",SS_CENTE
R | WS_CHILD,176,60,37,45
CONTROL "Please Answer the Following Questions",102,"STATIC",SS_CENTER | WS_CHILD,21,3
,183,12
```

```

CONTROL "WMPGRAPHIC",100,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,8,132,210,115
CONTROL "Beam Me Up Scotty !!! ***** I'm Through Collecting Data",101,"STATIC",SS_
CENTER | WS_CHILD,18,250,189,9
END

```

```
EXIT DIALOG 128,48,144,87
```

```
STYLE WS_POPUP | WS_CLIPSIBLINGS | WS_DLGFAME
```

```
BEGIN
```

```

CONTROL "YES",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,98,18,42,16
CONTROL "NO",2,"Button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,98,50,42,16
CONTROL "ARE YOU SURE YOU WANT TO EXIT?",100,"static",SS_CENTER | WS_CHILD,4,25,69,30
CONTROL "WMPGRAPHIC",102,"BUTTON",BS_OWNERDRAW | WS_CHILD | WS_DISABLED,78,18,16,16
CONTROL "WMPGRAPHIC",104,"BUTTON",BS_OWNERDRAW | WS_CHILD | WS_DISABLED,78,50,16,16
CONTROL "",105,"BUTTON",BS_GROUPBOX | WS_TABSTOP | WS_CHILD,4,18,70,39

```

```
END
```

```
statwopen DIALOG 164,25,107,120
```

```
STYLE WS_POPUP | WS_CLIPSIBLINGS | WS_DLGFAME
```

```
BEGIN
```

```

CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,39,99,30,14
CONTROL "System Unable to Open Stat*.fil For Writing",102,"static",SS_CENTER | WS_CHIL
D,28,38,55,27
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,24,1
0,16,16
CONTROL "See System Administrator",104,"static",SS_CENTER | WS_CHILD,28,72,53,19
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,67,1
0,16,16
END

```

```
statwrite DIALOG 164,25,107,120
```

```
STYLE WS_POPUP | WS_CLIPSIBLINGS | WS_DLGFAME
```

```
BEGIN
```

```

CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,42,94,30,14
CONTROL "System Unable to Write to Stat*.fil File",102,"static",SS_CENTER | WS_CHILD,2
0,35,70,19
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,24,1
0,16,16
CONTROL "See System Administrator",104,"static",SS_CENTER | WS_CHILD,28,63,53,19
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,67,1
0,16,16
END

```

```
BIT DIALOG 28,98,335,162
```

```
STYLE DS_MODALFRAME | WS_POPUP | WS_CLIPSIBLINGS | WS_CAPTION
```

```
CAPTION "BITMAPS"
```

```
FONT 10,"System"
```

```
BEGIN
```

```

CONTROL "Function",122,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,5,8,39,19
CONTROL "Function",123,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,5,29,39,19
CONTROL "Function",124,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,5,50,39,19
CONTROL "Function",125,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,5,71,39,19
CONTROL "Function",126,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,5,92,39,19
CONTROL "Function",127,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,5,113,39,19
CONTROL "Function",128,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,5,134,39,19
CONTROL "Function",129,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,61,8,39,19
CONTROL "Function",130,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,117,1,39,19
CONTROL "Function",131,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,118,44,39,19
CONTROL "Function",132,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,119,66,39,19
CONTROL "Function",133,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,61,29,39,19
CONTROL "Function",134,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,61,50,39,19
CONTROL "Function",135,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,61,71,39,19
CONTROL "Function",136,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,61,92,39,19
CONTROL "Function",137,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,61,113,39,19

```



```

CONTROL "Function",139,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,117,23,39,19
CONTROL "WMPGRAPHIC",141,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,178,1,32,32
CONTROL "WMPGRAPHIC",142,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,186,37,16,16
CONTROL "Function",143,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,177,57,39,19
CONTROL "Function",144,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,179,82,39,19
CONTROL "Func",145,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,248,12,35,18
CONTROL "Func",146,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,248,32,35,18
CONTROL "Func",147,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,248,52,35,18
CONTROL "Func",148,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,248,72,35,18
CONTROL "Func",149,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,248,92,35,18
CONTROL "Func",150,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,248,112,35,18
CONTROL "Func",151,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,248,132,35,18
CONTROL "ErrFunc",152,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,119,87,39,19
CONTROL "ErrFunc",153,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,118,108,39,19
CONTROL "ErrFunc",154,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,118,129,39,19
CONTROL "UserNumErr",155,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,179,103,39,19
CONTROL "Question",156,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,181,134,44,17
CONTROL "stat1",157,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,294,8,32,22
CONTROL "stat2",158,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,294,32,32,22
CONTROL "stat3",159,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,294,56,32,22
CONTROL "stat4",160,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,294,80,32,22

```

END

STATEMPTY DIALOG 93,0,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFRAME

BEGIN

```

CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,42,93,30,14
CONTROL " Stat*.fil File is Empty",102,"static",SS_CENTER | WS_CHILD,20,38,70,12
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,24,1
0,16,16
CONTROL "See System Administrator",104,"static",SS_CENTER | WS_CHILD,30,59,53,19
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,67,1
0,16,16

```

END

STATOPEN DIALOG 93,0,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGFRAME

BEGIN

```

CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,39,99,30,14
CONTROL "System Unable to Open Stat*.fil For Reading",102,"static",SS_CENTER | WS_CHIL
D,28,38,55,27
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,24,1
0,16,16
CONTROL "See System Administrator",104,"static",SS_CENTER | WS_CHILD,28,72,53,19
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,67,1
0,16,16

```

END

QINFO DIALOG 5,46,388,252

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS | WS\_CAPTION

CAPTION "Query Information and Possible Matches"

BEGIN

```

CONTROL "",IDLB_QMatch,"listbox",LBS_NOTIFY | LBS_USETABSTOPS | WS_TABSTOP | WS_CHILD
| WS_BORDER | WS_VSCROLL | WS_HSCROLL,8,104,371,119
CONTROL "Employer Detail",124,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,325,2,59,
15
CONTROL "Report Totals",ID_QRepTot,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,325,
19,59,15
CONTROL "Potential Blanket",IDBlanket,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,3
25,36,59,15
CONTROL "Close",ID_CLOSE,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,325,53,59,15
CONTROL " Addt'l Matches",ID_QAddMatch,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,
325,234,59,15
CONTROL "STATISTICS",156,"STATISTICS",WS_CHILD | BS_OWNERDRAW,7,81,373,44

```



```

CONTROL "MRN",132,"static",SS_CENTER | WS_CHILD,17,92,25,8
CONTROL "Wage Type",143,"static",SS_CENTER | WS_CHILD,73,83,21,17
CONTROL "FICA Wages",133,"static",SS_CENTER | WS_CHILD,97,83,23,17
CONTROL "FICA Tips",134,"static",SS_CENTER | WS_CHILD,136,83,19,17
CONTROL "FICA Tax W/H",135,"static",SS_CENTER | WS_CHILD,169,83,30,17
CONTROL "Wgs/Tips/ Other",139,"static",SS_CENTER | WS_CHILD,207,83,36,17
CONTROL "SSN",140,"static",SS_CENTER | WS_CHILD,255,92,30,8
CONTROL "NAME",136,"static",SS_CENTER | WS_CHILD,310,92,30,8
CONTROL "",ID_QEmprHeader,"STATIC",WS_CHILD | SS_LEFT,7,10,265,53
CONTROL "EMPLOYER BROWSE DATA",147,"STATIC",WS_CHILD | SS_LEFT,7,2,268,8
CONTROL "EMPLOYEE BROWSE DATA",149,"STATIC",WS_CHILD | SS_LEFT,3,73,381,8
CONTROL "",150,"STATIC",WS_CHILD | SS_LEFT,272,2,4,61
CONTROL "",151,"STATIC",WS_CHILD | SS_LEFT,3,2,4,61
CONTROL "",152,"STATIC",WS_CHILD | SS_LEFT,3,63,273,4
CONTROL "",153,"STATIC",WS_CHILD | SS_LEFT,3,225,380,4
CONTROL "",154,"STATIC",WS_CHILD | SS_LEFT,3,81,4,145
CONTROL "",155,"STATIC",WS_CHILD | SS_LEFT,380,81,4,148
CONTROL "Rep. No.",141,"static",SS_CENTER | WS_CHILD,57,83,16,17
CONTROL "***POSITION THE HIGHLIGHT BAR ON SELECTION AND EITHER DOUBLE-CLICK ON THE B
AR OR PRESS THE ENTER KEY TO DISPLAY THE EMPLOYEE DETAIL**",137,"STATIC",WS_CHILD | SS_LEF
T,4,234,226,16
CONTROL "Press Arrow to Select Different Report",158,"STATIC",SS_CENTER | WS_CHILD,284
,35,30,37
CONTROL "WMPGRAPHIC",ID_Change,"BUTTON",BS_OWNERDRAW | WS_CHILD,283,2,32,32
CONTROL "WMPGRAPHIC",159,"BUTTON",BS_OWNERDRAW | WS_TABSTOP | WS_CHILD,234,231,20,20
CONTROL " << Pres QP to Display ",160,"STATIC",WS_CHILD | SS_LEFT,256,234,63,8
CONTROL " << Query Parameters",162,"STATIC",WS_CHILD | SS_LEFT,256,242,63,8
END

BRREPORT DIALOG 5,46,388,252
STYLE DS_MODALFRAME | WS_POPUP | WS_CLIPSIBLINGS | WS_CAPTION
CAPTION "Browse Report"
BEGIN
CONTROL "",IDLB_BrMatch,"listbox",LBS_NOTIFY | LBS_USETABSTOPS | WS_TABSTOP | WS_CHILD
| WS_BORDER | WS_VSCROLL | WS_HSCROLL,8,104,371,119
CONTROL "Employer Detail",124,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,325,3,59,
18
CONTROL "Report Totals",ID_BrRepTot,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,325
,27,59,18
CONTROL "Close",ID_CLOSE,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,325,52,59,18
CONTROL " Addt'l Records",ID_BrAddRec,"BUTTON",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,3
25,232,59,18
CONTROL "",156,"STATIC",WS_CHILD | SS_LEFT,7,81,373,22
CONTROL "",ID_BrEmprHeader,"STATIC",WS_CHILD | SS_LEFT,7,10,265,53
CONTROL "MRN",132,"static",SS_CENTER | WS_CHILD,17,92,25,8
CONTROL "Rep. No.",141,"static",SS_CENTER | WS_CHILD,57,83,16,17
CONTROL "FICA Wages",133,"static",SS_CENTER | WS_CHILD,97,83,23,17
CONTROL "FICA Tips",134,"static",SS_CENTER | WS_CHILD,136,83,19,17
CONTROL "FICA Tax W/H",135,"static",SS_CENTER | WS_CHILD,169,83,30,17
CONTROL "Wgs/Tips/ Other",139,"static",SS_CENTER | WS_CHILD,207,83,36,17
CONTROL "SSN",140,"static",SS_CENTER | WS_CHILD,255,92,30,8
CONTROL "NAME",136,"static",SS_CENTER | WS_CHILD,310,92,30,8
CONTROL "***** POSITION THE HIGHLIGHT BAR ON SELECTION AND EITHER DOUBLE-CLICK ON TH
E BAR OR PRESS THE ENTER KEY TO DISPLAY THE EMPLOYEE DETAIL *****",137,"STATIC",SS_C
ENTER | WS_CHILD,4,234,314,16
CONTROL "EMPLOYER BROWSE DATA",147,"STATIC",WS_CHILD | SS_LEFT,7,2,268,8
CONTROL "EMPLOYEE BROWSE DATA",149,"STATIC",WS_CHILD | SS_LEFT,3,73,381,8
CONTROL "",150,"STATIC",WS_CHILD | SS_LEFT,272,2,4,61
CONTROL "",151,"STATIC",WS_CHILD | SS_LEFT,3,2,4,61
CONTROL "",152,"STATIC",WS_CHILD | SS_LEFT,3,63,273,4
CONTROL "",153,"STATIC",WS_CHILD | SS_LEFT,3,225,380,4
CONTROL "",154,"STATIC",WS_CHILD | SS_LEFT,3,81,4,145
CONTROL "",155,"STATIC",WS_CHILD | SS_LEFT,380,81,4,148
CONTROL "Wage Type",143,"static",SS_CENTER | WS_CHILD,73,83,21,17

```

END

BINFO DIALOG 5,46,388,252

STYLE DS\_MODALFRAME | WS\_POPUP | WS\_CLIPSIBLINGS | WS\_CAPTION

CAPTION "Blanket Information "

BEGIN

```
CONTROL "",IDLB_BMatch,"listbox",LBS_NOTIFY | LBS_USETABSTOPS | WS_TABSTOP | WS_CHILD
| WS_BORDER | WS_VSCROLL | WS_HSCROLL,8,104,371,119
CONTROL "Employer Detail",124,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,325,1,59,
15
CONTROL "Report Totals",ID_BRepTot,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,325,
18,59,15
CONTROL "Print Blanket",IDPRINTB,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,325,35
,59,15
CONTROL "Close",ID_CLOSE,"button",WS_TABSTOP | WS_CHILD | BS_PUSHBUTTON,325,52,59,15
CONTROL "",156,"STATIC",WS_CHILD | SS_LEFT,7,81,373,22
CONTROL "FICA Wages",133,"static",SS_CENTER | WS_CHILD,97,83,23,17
CONTROL "FICA Tips",134,"static",SS_CENTER | WS_CHILD,136,83,19,17
CONTROL "FICA Tax W/H",135,"static",SS_CENTER | WS_CHILD,169,83,30,17
CONTROL "Wgs/Tips/ Other",139,"static",SS_CENTER | WS_CHILD,207,83,36,17
CONTROL "SSN",140,"static",SS_CENTER | WS_CHILD,255,92,30,8
CONTROL "NAME",136,"static",SS_CENTER | WS_CHILD,310,92,30,8
CONTROL "***** POSITION THE HIGHLIGHT BAR ON SELECTION AND EITHER DOUBLE-CLICK ON TH
E BAR OR PRESS THE ENTER KEY TO DISPLAY THE EMPLOYEE DETAIL *****",137,"STATIC",SS_C
ENTER | WS_CHILD,35,233,314,16
CONTROL "",ID_BEmprHeader,"STATIC",WS_CHILD | SS_LEFT,7,10,265,53
CONTROL "Rep. No.",141,"static",SS_CENTER | WS_CHILD,57,83,16,17
CONTROL "Wage Type",143,"static",SS_CENTER | WS_CHILD,73,83,21,17
CONTROL "EMPLOYER BROWSE DATA",147,"STATIC",WS_CHILD | SS_LEFT,7,2,268,8
CONTROL "EMPLOYEE BROWSE DATA",149,"STATIC",WS_CHILD | SS_LEFT,3,73,381,8
CONTROL "",150,"STATIC",WS_CHILD | SS_LEFT,272,2,4,61
CONTROL "",151,"STATIC",WS_CHILD | SS_LEFT,3,2,4,61
CONTROL "",152,"STATIC",WS_CHILD | SS_LEFT,3,63,273,4
CONTROL "",153,"STATIC",WS_CHILD | SS_LEFT,3,225,380,4
CONTROL "",154,"STATIC",WS_CHILD | SS_LEFT,3,81,4,145
CONTROL "",155,"STATIC",WS_CHILD | SS_LEFT,380,81,4,148
CONTROL "MRN",132,"static",SS_CENTER | WS_CHILD,17,92,25,8
```

END

DETAILTXT DIALOG 113,25,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGMFRAME

BEGIN

```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,37,99,30,14
CONTROL "System Unable to Open detl*.txt File",102,"static",SS_CENTER | WS_CHILD,8,39,
92,19
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,24,1
0,16,16
CONTROL "See System Administrator",104,"static",SS_CENTER | WS_CHILD,15,67,77,19
CONTROL "WMPGRAPHIC",105,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,67,1
0,16,16
END
```

EINORSEQ DIALOG 148,25,131,150

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGMFRAME

BEGIN

```
CONTROL "OK",1,"Button",BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,52,131,30,14
CONTROL "No Report Exists for the Requested EIN or Report Number",102,"static",SS_CENT
ER | WS_CHILD,11,30,111,18
CONTROL "WMPGRAPHIC",103,"BUTTON",BS_OWNERDRAW | WS_GROUP | WS_TABSTOP | WS_CHILD,41,6
,16,16
CONTROL "Press OK, Hit Return or ESC, and Re-Enter the EIN or Report Number",104,"stat
ic",SS_CENTER | WS_CHILD,17,53,96,27
CONTROL "WAIT!",106,"STATIC",WS_CHILD | SS_LEFT,71,11,27,10
CONTROL "NOTE: This Box Also Appears When You Try to Print a File Larger Than 5000 Em
```



ployees. See the System Administrator to Print Large Reports.",107,"STATIC",SS\_CENTER | WS\_CHILD,14,83,104,42  
END

HEADERTXT DIALOG 113,25,107,120

STYLE WS\_POPUP | WS\_CLIPSIBLINGS | WS\_DLGBFRAME

BEGIN

CONTROL "OK",1,"Button",BS\_DEFPUSHBUTTON | WS\_TABSTOP | WS\_CHILD,37,99,30,14

CONTROL "System Unable to Open hdr\*.txt File",102,"static",SS\_CENTER | WS\_CHILD,8,39,92,19

CONTROL "WMPGRAPHIC",103,"BUTTON",BS\_OWNERDRAW | WS\_GROUP | WS\_TABSTOP | WS\_CHILD,24,10,16,16

CONTROL "See System Administrator",104,"static",SS\_CENTER | WS\_CHILD,15,67,77,19

CONTROL "WMPGRAPHIC",105,"BUTTON",BS\_OWNERDRAW | WS\_GROUP | WS\_TABSTOP | WS\_CHILD,67,10,16,16

END



```
; Filename: EAMAT42.DEF
; "EAMAT42" Generated by WindowsMAKER Professional.
; Author: Laura L. Downey

;
; *****
; Do not add code here.
;
; This file is maintained by WindowsMAKER Professional.
; As you make changes in your application using WindowsMAKER Professional,
; this file is automatically updated, therefore you never modify this file.
;
;
; For more information,
; see the section "How code is generated" in the documentation.
;
; *****
;
```

```
NAME          EAMAT42
DESCRIPTION    'EAMATE generated by WindowsMAKER Professional'
EXETYPE        WINDOWS
STUB           'WINSTUB.EXE'
HEAPSIZE       10248
STACKSIZE      8192
DATA           MOVEABLE MULTIPLE
CODE           MOVEABLE DISCARDABLE PRELOAD
SEGMENTS
```

```
    _USERCODE  MOVEABLE DISCARDABLE PRELOAD
    _EAMAT42   MOVEABLE DISCARDABLE PRELOAD
    _SERVICE  MOVEABLE DISCARDABLE PRELOAD
    _CUSTOM    MOVEABLE DISCARDABLE PRELOAD
    _ERROR     MOVEABLE DISCARDABLE PRELOAD
    _PRINT     MOVEABLE DISCARDABLE PRELOAD
    _STAT      MOVEABLE DISCARDABLE PRELOAD
    _TEXT      MOVEABLE DISCARDABLE PRELOAD
```

## EXPORTS

```
BLDMainWndProc
BLD_QUERYDlgProc
BLD_MAINC1Proc
BLD_PrintDlgProc
BLD_OKDlgProc
BLD_FunctionDlgProc
BLD_Function2DlgProc
BLD_Function6DlgProc
BLD_HeaderDetailDlgProc
BLD_EmployeeDetailDlgProc
BLD_BrowseReportDlgProc
BLD_ReportStatisticsDlgProc
BLD_ReportDlgProc
BLD_BrowseEntryDlgProc
BLD_ReportTotalsDlgProc
BLD_EINErrDlgProc
BLD_QueryMessageDlgProc
BLD_NMSGDlgProc
BLD_QueryErrDlgProc
BLD_BlanketErrDlgProc
BLD_PWErrDlgProc
BLD_Year_ErrDlgProc
BLD_SysErrDlgProc
BLD_BlankErrDlgProc
BLD_MissingFileDlgProc
BLD_QueryTxtDlgProc
```

BLD\_ErrorFileDlgProc  
BLD\_NULLPtrDlgProc  
BLD\_DataErrDlgProc  
BLD\_DFileErrDlgProc  
BLD\_NoMoreMatchesDlgProc  
BLD\_SequenceDlgProc  
BLD\_MRNErrDlgProc  
BLD\_MatchErrDlgProc  
BLD\_PrintBlanketDlgProc  
BLD\_PrintEmpDetailDlgProc  
BLD\_PrintHeaderDetailDlgProc  
BLD\_Function5DlgProc  
BLD\_TotNowPrintDlgProc  
BLD\_GetNumCopyDlgProc  
BLD\_qparamDlgProc  
BLD\_EINorSeqErrDlgProc  
BLD\_SeqErrDlgProc  
BLD\_HFileDlgProc  
BLD\_UserNumErrDlgProc  
BLD\_questDlgProc  
BLD\_StatOpenDlgProc  
BLD\_StatEmptyDlgProc  
BLD\_StatWOpenDlgProc  
BLD\_StatWriteErrDlgProc

eamat42.lnk

Wed Mar 2 14:13:49 1994

1

USERCODE EAMAT42 SERVICE CUSTOM ERROR PRINT STAT

EAMAT42.EXE

/map /CO /align:16 /NOD

LIBW MLIBCEW tklibw mfedit

EAMAT42.DEF



```
// Filename: EAMAT42.RC
// "EAMAT42" Generated by WindowsMAKER Professional.
// Author: Laura L. Downey

//
// *****
// Do not add code here.
//
// This file is maintained by WindowsMAKER Professional.
// As you make changes in your application using WindowsMAKER Professional,
// this file is automatically updated, therefore you never modify this file.
//
//
// For more information,
// see the section "How code is generated" in the documentation.
//
// *****
//

#define THISISBLDRC

#include "WINDOWS.H"
#include "GENERIC.H"

#include "DIALOG.DLG"

ARCADE    BITMAP    ARCADE.BMP
ARCHES    BITMAP    ARCHES.BMP
ARGYLE    BITMAP    ARGYLE.BMP
BOXES     BITMAP    BOXES.BMP
CASTLE    BITMAP    CASTLE.BMP
EGYPT     BITMAP    EGYPT.BMP
QUESTION  BITMAP    QUESTION.BMP
RIVETS    BITMAP    RIVETS.BMP
THATCH    BITMAP    THATCH.BMP
WELCOME   BITMAP    WELCOME.BMP
ZIGZAG    BITMAP    ZIGZAG.BMP
CONTINUE  BITMAP    CONTINUE.BMP
LINE      BITMAP    LINE.BMP
OUTLINE   BITMAP    OUTLINE.BMP
QPARAM    BITMAP    QPARAM.BMP
BRANCH    BITMAP    BRANCH.BMP
EYE       BITMAP    EYE.BMP
CH         BITMAP    CH.BMP
ARR        BITMAP    ARR.BMP
QUEST     BITMAP    QUEST.BMP
X          BITMAP    X.BMP
PARAM     BITMAP    PARAM.BMP
BLKHOLE   BITMAP    BLKHOLE.BMP
ENTERPRZ  BITMAP    ENTERPRZ.BMP

CHART1    ICON      CHART1.ICO
CHART3    ICON      CHART3.ICO
CHART4    ICON      CHART4.ICO
CHART5    ICON      CHART5.ICO
FELLIPSE  ICON      FELLIPSE.ICO
FFIND     ICON      FFIND.ICO
NOTE      ICON      NOTE.ICO
STOP      ICON      STOP.ICO
TERMINAL  ICON      TERMINAL.ICO
TTT       ICON      TTT.ICO
```

PC1	ICON	PC1.ICO
NOTE1	ICON	NOTE1.ICO
NOTE2	ICON	NOTE2.ICO
EDIT1	ICON	EDIT1.ICO
EDIT2	ICON	EDIT2.ICO
EXIT1	ICON	EXIT1.ICO
EXIT2	ICON	EXIT2.ICO

```
// *****  
//      Resource code for error message strings  
// *****
```

## STRINGTABLE

BEGIN

BLD_CannotRun	"Cannot run "
BLD_CannotCreate	"Cannot create dialog box "
BLD_CannotLoadMenu	"Cannot load menu "
BLD_CannotLoadIcon	"Cannot load icon "
BLD_CannotLoadBitmap	"Cannot load bitmap "
BLD_CannotCreateWindow	"Cannot create window "

END

```
// Filename: EAMAT42.WMC
// "EAMAT42" Generated by WindowsMAKER Professional.
// Author: Laura L. Downey

//
// *****
// Do not add code here. Add code in the .C file.
//
// This file is maintained by WindowsMAKER Professional.
// As you make changes in your application using WindowsMAKER Professional,
// this file is automatically updated, therefore you never modify this file.
//
//
// For more information,
// see the section "How code is generated" in the documentation.
//
// *****

// *****
// GLOBAL VARIABLES
// *****

HINSTANCE hInst          = 0;    // Handle to instance.
HWND      MainhWnd       = 0;    // Handle to main window.
UINT      wBLDWindowType = 0;    // Registered Message for toolbars
UINT      wHelpMessage   = 0;    // Registered Message for Help
BOOL      bHelpSupport    = FALSE; // Controlling Help Support
DWORD     dwDialogProp    = 0;    // Controlling Dialog Box Color
BOOL      b256Color       = TRUE; // Controlling bitmap drawing
HINSTANCE hBMPInst       = 0;    // Handle to instance for bitmaps
HWND      hClient         = 0;    // Handle to window in client area.
DLGPROC   lpClient        = 0L;   // Function for window in client area.

HWND      hMDIClient      = 0;    // Handle to client window for MDI.
UINT      idMDIFirstChild = 100;  // ID to first child for MDI.

HWND MAINhDlg = 0;
DLGPROC MAINlpProc = 0L;

// *****
// FUNCTIONS FOR INITIALIZATION AND EXIT OF APPLICATION
// *****

BOOL BLDInitApplication(HINSTANCE hInst, HINSTANCE hPrev, int *pCmdShow, LPSTR lpCmd)
{
    wBLDWindowType = RegisterWindowMessage("BLDToolbarMessage");
    wHelpMessage = RegisterWindowMessage("BLDHelpMessage");
    return BLD_ApplicationAppInit(hInst, hPrev, pCmdShow, lpCmd);
}

// Registers the classes for all windows
BOOL BLDRegisterClass(HINSTANCE hInstance)
{
    BOOL bReturn;

    bReturn = BLDMainRegClass(hInstance);

    return bReturn;
}
```



```
}

// Creates windows at Initialization
HWND BLDCreateWindow(HINSTANCE hInstance)
{
    return BLDMainCreateWnd();
}

// Called just before entering message loop
BOOL BLDInitMainMenu(HWND hWnd)
{
    return TRUE;
}

BOOL BLDExitApplication()          // Called just before exit of application
{
#ifdef WIN32
    if (TRUE)
#else
    if (GetModuleUsage(hInst) == 1)
#endif
    {
        BLDMainExitClass();
    }

    return TRUE;
}

// *****
//          PROCESSES KEYBOARD ACCELERATORS
//          AND MODELESS DIALOG BOX KEY INPUT
// *****

BOOL BLDKeyTranslation(MSG *pMsg)
{
    // Translates keystrokes for toolbars and client area controls
    if (BLDIsClientDlgDialogMessage(pMsg))
        return TRUE;
    // Translates keystrokes for modeless dialog box
    if (MAINhDlg && IsDialogMessage(MAINhDlg, pMsg))
        return TRUE;
    // Translates keystrokes so client area works as a dialog box
    if (hClient && IsDialogMessage(hClient, pMsg))
        return TRUE;
    return FALSE; // No special key input
}

// Registers the class for the main window
BOOL BLDMainRegClassDef(HINSTANCE hInstance)
{
    WNDCLASS    WndClass;

    WndClass.style          = CS_DBLCLKS;
    WndClass.lpfnWndProc    = BLDMainWndProc;
    WndClass.cbClsExtra     = 0;
    WndClass.cbWndExtra     = 0;
    WndClass.hInstance      = hInstance;
```

```
WndClass.hIcon          = LoadIcon(hInstance,"PC1");
WndClass.hCursor        = LoadCursor(NULL,IDC_ARROW);
WndClass.hbrBackground  = (HBRUSH)(COLOR_WINDOW+1);
WndClass.lpszMenuName    = NULL;
WndClass.lpszClassName  = "BLDEAMAT42";
```

```
return RegisterClass(&WndClass);
}
```

```
BOOL BLDMainExitClassDef()
```

```
{
    return TRUE;
}
```

```
// Creates the main window
```

```
HWND BLDMainCreateWndDef()
```

```
{
    HWND    hWnd;                // window handle
    int     coordinate[4];       // Coordinates of main window
```

```
    coordinate[0]=-4;
    coordinate[1]=-4;
    coordinate[2]=648;
    coordinate[3]=488;
```

```
    hWnd = CreateWindowEx(0, // window Extended style
        "BLDEAMAT42", // window class registered earlier
        "EAMATE V4.2", // window caption
        WS_THICKFRAME | WS_CLIPCHILDREN | WS_MINIMIZEBOX | WS_OVERLAPPED,
```

```
// window style
        coordinate[0], // x position
        coordinate[1], // y position
        coordinate[2], // width
        coordinate[3], // height
        0, // parent handle
        0, // menu or child ID
        hInst, // instance
        (LPSTR)NULL); // additional info
```

```
    return hWnd;
}
```

```
// *****
//          CUSTOM MESSAGE PROCESSING FOR MAIN WINDOW
// *****
```

```
LRESULT BLDDefWindowProc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
```

```
{
    LRESULT    lReturn;
```

```
    lReturn    = 0;
```

```
    if(BLDWndMsgFilter(hWnd,message,wParam,lParam,0,&lReturn))
        return lReturn;
```

```
    switch (message)
    {
```

```
        case WM_SIZE:
```

```
        BLDSIZEToolBars(hWnd,message,(int)wParam,(int)LOWORD(lParam),(int)HIWORD(lParam),F
ELSE);
```

```
        break;
```

```
case WM_CREATE:
```

```
    MainhWnd=hWnd;
```

```
    BLD_MAINClFunc(hWnd,message,wParam,lParam);
```

```
    break;
```

```
case WM_MOVE:
```

```
    BLDClientMove(hWnd);
```

```
    break;
```

```
case WM_SETFOCUS:
```

```
    BLDSetClientFocus(hWnd);
```

```
    break;
```

```
default:
```

```
    // Pass on message for default processing by Windows
```

```
    return DefWindowProc(hWnd,message,wParam,lParam);
```

```
    break;
```

```
    }
```

```
return FALSE; // Returns FALSE if not processed by Windows
```

```
}
```

```
// *****
```

```
//          PROCESSES ALL MENU ITEM SELECTIONS
```

```
// *****
```

```
BOOL BLDMenuCommand(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
```

```
{
```

```
    switch(LOWORD(wParam))
```

```
    {
```

```
    // Processing of linked menu items in menu: GENERIC
```

```
default:
```

```
    return FALSE; // Not processed by this function.
```

```
    }
```

```
return TRUE; // Processed by this function.
```

```
}
```

```
// *****
```

```
//          PROCESSES HELP FOR MENU ITEMS
```

```
// *****
```

```
BOOL BLDMenuHelp(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
```

```
{
```

```
    switch(LOWORD(wParam))
```

```
    {
```

```
default:
```

```
    PostMessage(hWnd,wHelpMessage,0,0L);
```

```
    return FALSE; // Processed as help for window.
```

```
    }
```

```
return TRUE; // Processed by this function.
```

```
}
```

```
// *****
```

```
//          WINDOW & DIALOG BOX COMMON PROCESSING
```

```
// *****
```



```
BOOL BLDWndMsgFilter(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam,
                    DWORD dwHelpId,LRESULT *plRetval)
{
    return BLDHelpFilter(hWnd,message,wParam,lParam,dwHelpId,plRetval,FALSE);
}
```

```
BOOL BLDDlgMsgFilter(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam,
                    int iWndType,DWORD dwHelpId,BOOL *pbRetval)
{
    LRESULT      lReturn;
    BOOL         bRet;

    lReturn      = 0L;

    if (message==wBLDWindowType)
    {
        *((LPINT)lParam)=iWndType;
        *pbRetval      =TRUE;
        return TRUE;
    }
    bRet = BLDHelpFilter(hDlg,message,wParam,lParam,dwHelpId,&lReturn,TRUE);
    *pbRetval = (BOOL)lReturn;
    return bRet;
}
```

```
BOOL BLDGetHelpFileName(char *szWinHelpFile)
{
    OFSTRUCT of;
    char path[BLD_MAXPATH];
    int len;
    char *pStr;

    GetModuleFileName(hInst,path,sizeof(path));
    len=lstrlen(path);
    for (pStr=&path[len-1];pStr>path;--pStr)
    {
        if (*pStr=='\\')
        {
            pStr++;
            lstrcpy(pStr,"EAMAT42.HLP");
            if (OpenFile(path,&of,OF_EXIST)!=-1)
            {
                lstrcpy(szWinHelpFile,path);
                return TRUE;
            }
            else
                goto RET_DEF;
        }
    }
RET_DEF:
    lstrcpy(szWinHelpFile,"EAMAT42.HLP");
    return TRUE;
}
```

```
// *****
//      FUNCTION FOR SWITCHING MENU SET
// *****
```

```
BOOL BLDSwitchMenu(HWND hWnd, char *pTemplateName)
{
    HMENU      hMenu1, hMenu, hSubMenu;
    DWORD      style;

    hSubMenu = 0;

    style = GetWindowLong(hWnd, GWL_STYLE);
    if((style & WS_CHILD) == WS_CHILD) // Called from control in main window?
    {
        hWnd = GetParent(hWnd);
        if (!hWnd)
            return FALSE;
        style = GetWindowLong(hWnd, GWL_STYLE);
        if((style & WS_CHILD) == WS_CHILD) // No menu in a WS_CHILD window.
            return FALSE;
    }
    if((style & WS_CAPTION) != WS_CAPTION) // No menu if no caption.
        return FALSE;

    hMenu1 = GetMenu(hWnd);
    hMenu = BLDLoadMenu(hWnd, pTemplateName, &hSubMenu);
    if (!hMenu)
    {
        BLDDisplayMessage(hWnd, BLD_CannotLoadMenu, pTemplateName,
                           MB_OK | MB_ICONASTERISK);
        return FALSE;
    }

    if (!SetMenu(hWnd, hMenu))
        return FALSE;
    if (hMenu1)
        DestroyMenu(hMenu1);

    DrawMenuBar(hWnd);
    return TRUE;
}

// Code to load menu and add bitmaps and accelerators
HMENU BLDLoadMenu(HWND hWnd, char *pTemplateName, HMENU *phSubMenu)
{
    HMENU      hMenu;

    *phSubMenu = 0;
    hMenu = LoadMenu(hInst, pTemplateName);

    if (!hMenu)
        return FALSE;

    return hMenu;
}
```

```
// Filename: ERROR.WMC
// "EAMAT42" Generated by WindowsMAKER Professional.
// Author: Laura L. Downey

//
// *****
// Do not add code here. Add code in the .C file.
//
// This file is maintained by WindowsMAKER Professional.
// As you make changes in your application using WindowsMAKER Professional,
// this file is automatically updated, therefore you never modify this file.
//
//
// For more information,
// see the section "How code is generated" in the documentation.
//
// *****

// *****
// Modal Dialog Box: EINERROR
// *****

// Startup procedure for modal dialog box
int BLD_EINErrDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC lpProc;
    int ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_EINErrDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"EINERROR"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"EINERROR"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_EINErrDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL bRet;

    bRet = FALSE; // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {
        case WM_INITDIALOG:
            bRet = TRUE; // Default return for WM_INITDIALOG is TRUE
            break;

        case WM_COMMAND:
            {
                HWND hCtrl;
```



```
// Extracting data from message
hCtrl      = (HWND) (UINT) lParam;
if (!hCtrl)      // Menu input or CR
{
    if (BLDMenuCommand(hDlg, message, wParam, lParam))
        return TRUE;
}
break;

case WM_DRAWITEM:
{
    LPDRAWITEMSTRUCT lpDrawItem;

    lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
    switch (lpDrawItem->CtlID)
    {
        case 103:
            if (lpDrawItem->itemAction == ODA_DRAWENTIRE)
                if (lpDrawItem->CtlType == ODT_BUTTON)
                    BLDDrawIcon(lpDrawItem, "NOTE");
            return TRUE;
            break;
        default:
            if (BLDDrawItem(hDlg, lpDrawItem))
                return TRUE;
            break;
    }
    break;

default:
    break;
}
return bRet;      // No explicit return - return default
}

// *****
//          Modal Dialog Box: QMSG
// *****

// Startup procedure for modal dialog box
int BLD_QueryMessageDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC) MakeProcInstance((FARPROC) BLD_QueryMessageDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR) (szDlgName ? szDlgName : "QMSG"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC) lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName ? szDlgName : "QMSG"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_QueryMessageDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL          bRet;

```

```
bRet      = FALSE;    // Default return value if not processed

if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
    return bRet;

switch(message)
{

case WM_INITDIALOG:
    bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
    break;

case WM_COMMAND:
    {
        WORD      wId;
        WORD      notification;
        HWND      hCtrl;

        // Extracting data from message
        wId      = LOWORD(wParam);
        hCtrl     = (HWND)(UINT)lParam;

#ifdef WIN32
        notification = HIWORD(wParam);
#else
        notification = HIWORD(lParam);
#endif

        if(!hCtrl)            // Menu input or CR
        {
            if (BLDMenuCommand(hDlg,message,wParam,lParam))
                return TRUE;
        }

        switch(wId)
        {
            case 2:
                switch(notification)
                {
                    case BN_CLICKED:
                        EndDialog(hDlg,2);
                        return TRUE;
                        break;
                    default:
                        break;
                }
                break;
            default:
                break;
        }
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        switch(lpDrawItem->CtlID)
        {
            case 101:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"BOXES",TRUE);
                return TRUE;
                break;
            case 102:

```

```
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDrawBitmap(lpDrawItem,"BOXES",TRUE);
        return TRUE;
        break;
    default:
        if (BLDDrawItem(hDlg,lpDrawItem))
            return TRUE;
        break;
    }
    break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//           Modal Dialog Box: NMSG
// *****

// Startup procedure for modal dialog box
int BLD_NMSGDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int         ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_NMSGDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"NMSG"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue==-1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"NMSG"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_NMSGDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND        hCtrl;

            // Extracting data from message
            hCtrl        = (HWND)(UINT)lParam;
        }
    }
}
```



```
        if(!hCtrl)                // Menu input or CR
        {
            if (BLDMenuCommand(hDlg,message,wParam,lParam))
                return TRUE;
        }
        break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        switch(lpDrawItem->CtlID)
        {
            case 101:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"FELLIPSE");
                return TRUE;
                break;
            case 102:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"FELLIPSE");
                return TRUE;
                break;
            default:
                if(BLDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

default:
    break;
}
return bRet;                // No explicit return - return default
}

// *****
//                               Modal Dialog Box: QUERYERROR
// *****

// Startup procedure for modal dialog box
int BLD_QueryErrDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_QueryErrDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"QUERYERROR"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"QUERYERROR"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
```

```
BOOL BLD_QueryErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL      bRet;

    bRet      = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND      hCtrl;

            // Extracting data from message
            hCtrl      = (HWND)(UINT)lParam;
            if(!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand(hDlg,message,wParam,lParam))
                    return TRUE;
            }
        }
        break;

    case WM_DRAWITEM:
        {
            LPDRAWITEMSTRUCT lpDrawItem;

            lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
            switch(lpDrawItem->CtlID)
            {
                case 103:
                    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                        if (lpDrawItem->CtlType==ODT_BUTTON)
                            BLDDrawIcon(lpDrawItem,"STOP");
                    return TRUE;
                    break;
                case 105:
                    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                        if (lpDrawItem->CtlType==ODT_BUTTON)
                            BLDDrawIcon(lpDrawItem,"STOP");
                    return TRUE;
                    break;
                default:
                    if (BLDDrawItem(hDlg,lpDrawItem))
                        return TRUE;
                    break;
            }
        }
        break;

    default:
        break;
    }
    return bRet;    // No explicit return - return default
}
```

```
// *****
//                               Modal Dialog Box: BLANKETERROR
// *****

// Startup procedure for modal dialog box
int BLD_BlanketErrDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_BlanketErrDlgProc, hWnd);
    ReturnValue = DialogBox(hWnd, (LPSTR)(szDlgName?szDlgName:"BLANKETERROR"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"BLANKETERROR"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_BlanketErrDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL          bRet;

    bRet          = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {
        case WM_INITDIALOG:
            bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
            break;

        case WM_COMMAND:
            {
                HWND          hCtrl;

                // Extracting data from message
                hCtrl          = (HWND)(UINT)lParam;
                if(!hCtrl)      // Menu input or CR
                {
                    if (BLDMenuCommand(hDlg, message, wParam, lParam))
                        return TRUE;
                }
            }
            break;

        case WM_DRAWITEM:
            {
                LPDRAWITEMSTRUCT lpDrawItem;

                lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
                switch(lpDrawItem->CtlID)
                {
                    case 103:
                        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                            if (lpDrawItem->CtlType==ODT_BUTTON)
                                BLDDrawIcon(lpDrawItem, "STOP");
                        return TRUE;
                    }
            }
    }
}
```



```
        break;
    case 105:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDDrawIcon(lpDrawItem,"STOP");
        return TRUE;
        break;
    default:
        if (BLDDDrawItem(hDlg,lpDrawItem))
            return TRUE;
        break;
    }
    break;

default:
    break;
}
return bRet;                // No explicit return - return default
}

// *****
//          Modal Dialog Box: PERROR
// *****

// Startup procedure for modal dialog box
int BLD_PWErrDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_PWErrDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"PERROR"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue==-1)
        BLDDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"PERROR"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_PWErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND        hCtrl;
```

```

    // Extracting data from message
    hCtrl      = (HWND) (UINT) lParam;
    if (!hCtrl)           // Menu input or CR
    {
        if (BLDMenuCommand(hDlg, message, wParam, lParam))
            return TRUE;
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
        switch (lpDrawItem->CtlID)
        {
            case 103:
                if (lpDrawItem->itemAction == ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType == ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem, "NOTE");
                return TRUE;
                break;
            default:
                if (BLDDrawItem(hDlg, lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

default:
    break;
}

return bRet;           // No explicit return - return default
}

// *****
//           Modal Dialog Box: YEAR_ERR
// *****

// Startup procedure for modal dialog box
int BLD_Year_ErrDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC) MakeProcInstance((FARPROC) BLD_Year_ErrDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR) (szDlgName?szDlgName:"YEAR_ERR"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC) lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"YEAR_ERR"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_Year_ErrDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL        bRet;

```

```
bRet      = FALSE;    // Default return value if not processed

if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
    return bRet;

switch(message)
{

case WM_INITDIALOG:
    bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
    break;

case WM_COMMAND:
    {
        HWND      hCtrl;

        // Extracting data from message
        hCtrl      = (HWND) (UINT) lParam;
        if (!hCtrl)    // Menu input or CR
        {
            if (BLDMenuCommand(hDlg,message,wParam,lParam))
                return TRUE;
        }
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
        switch(lpDrawItem->CtlID)
        {
            case 101:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDDrawIcon(lpDrawItem,"NOTE");
                return TRUE;
                break;
            default:
                if (BLDDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

default:
    break;
}

return bRet;    // No explicit return - return default
}

// *****
//          Modal Dialog Box: SYSERR
// *****

// Startup procedure for modal dialog box
int BLD_SysErrDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;
```



```
lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_SysErrDlgProc,hInst);
ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"SYSERR"),
                        hWnd,lpProc);
FreeProcInstance((FARPROC)lpProc);
if (ReturnValue==-1)
    BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"SYSERR"),
                      MB_OK | MB_ICONHAND);
return ReturnValue;
}
```

// Default dialog box procedure

```
BOOL BLD_SysErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL      bRet;

    bRet      = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND      hCtrl;

            // Extracting data from message
            hCtrl      = (HWND)(UINT)lParam;
            if(!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand(hDlg,message,wParam,lParam))
                    return TRUE;
            }
        }
        break;

    case WM_DRAWITEM:
        {
            LPDRAWITEMSTRUCT lpDrawItem;

            lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
            switch(lpDrawItem->CtlID)
            {
                case 103:
                    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                        if (lpDrawItem->CtlType==ODT_BUTTON)
                            BLDDrawIcon(lpDrawItem,"STOP");
                    return TRUE;
                    break;

                case 105:
                    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                        if (lpDrawItem->CtlType==ODT_BUTTON)
                            BLDDrawIcon(lpDrawItem,"STOP");
                    return TRUE;
                    break;

                default:
                    if (BLDDrawItem(hDlg,lpDrawItem))
                        return TRUE;
            }
        }
    }
}
```

```
        break;
    }
    break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//          Modal Dialog Box: BLANKERR
// *****

// Startup procedure for modal dialog box
int BLD_BlankErrDlgFuncDef (HWND hWnd, char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_BlankErrDlgProc, hWnd);
    ReturnValue = DialogBox(hWnd, (LPSTR)(szDlgName?szDlgName:"BLANKERR"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"BLANKERR"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_BlankErrDlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL          bRet;

    bRet          = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch (message)
    {
    case WM_INITDIALOG:
        bRet          = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND      hCtrl;

            // Extracting data from message
            hCtrl      = (HWND)(UINT)lParam;
            if (!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand(hDlg, message, wParam, lParam))
                    return TRUE;
            }
        }
        break;
    }
```

```

    case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        switch(lpDrawItem->CtlID)
        {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            default:
                if (BLDDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//           Modal Dialog Box: MISSINGFILE
// *****

// Startup procedure for modal dialog box
int BLD_MissingFileDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_MissingFileDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"MISSINGFILE"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue==-1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"MISSINGFILE"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_MissingFileDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDLgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;
}

```



```
switch(message)
{
case WM_INITDIALOG:
    bRet      = TRUE;      // Default return for WM_INITDIALOG is TRUE
    break;

case WM_COMMAND:
    {
        HWND      hCtrl;

        // Extracting data from message
        hCtrl      = (HWND)(UINT)lParam;
        if(!hCtrl)      // Menu input or CR
        {
            if (BLDMenuCommand(hDlg,message,wParam,lParam))
                return TRUE;
        }
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        switch(lpDrawItem->CtlID)
        {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            default:
                if (BLDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

default:
    break;
}

return bRet;      // No explicit return - return default
}

// *****
//          Modal Dialog Box: QUERYTXT
// *****

// Startup procedure for modal dialog box
int BLD_QueryTxtDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC      lpProc;
```

```
int      ReturnValue;

lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_QueryTxtDlgProc,hInst);
ReturnValue = DialogBox(hInst, (LPSTR) (szDlgName?szDlgName:"QUERYTXT"),
                        hWnd,lpProc);
FreeProcInstance((FARPROC)lpProc);
if (ReturnValue== -1)
    BLDDisplayMessage(hWnd,BLD_CannotCreate, (szDlgName?szDlgName:"QUERYTXT"),
                      MB_OK | MB_ICONHAND);
return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_QueryTxtDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL      bRet;

    bRet      = FALSE;    // Default return value if not processed

    if(BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND      hCtrl;

            // Extracting data from message
            hCtrl      = (HWND) (UINT)lParam;
            if(!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand(hDlg,message,wParam,lParam))
                    return TRUE;
            }
        }
        break;

    case WM_DRAWITEM:
        {
            LPDRAWITEMSTRUCT lpDrawItem;

            lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
            switch(lpDrawItem->CtlID)
            {
                case 103:
                    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                        if (lpDrawItem->CtlType==ODT_BUTTON)
                            BLDDrawIcon(lpDrawItem,"STOP");
                    return TRUE;
                    break;
                case 105:
                    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                        if (lpDrawItem->CtlType==ODT_BUTTON)
                            BLDDrawIcon(lpDrawItem,"STOP");
                    return TRUE;
                    break;
                default:

```

```
        if (BLDDDrawItem(hDlg,lpDrawItem))
            return TRUE;
        break;
    }
    break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//          Modal Dialog Box: ERRORFILE
// *****

// Startup procedure for modal dialog box
int BLD_ErrorFileDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_ErrorFileDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"ERRORFILE"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"ERRORFILE"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_ErrorFileDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND        hCtrl;

            // Extracting data from message
            hCtrl        = (HWND)(UINT)lParam;
            if (!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand(hDlg,message,wParam,lParam))
                    return TRUE;
            }
        }
    }
}
```



```

        break;

    case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        switch(lpDrawItem->CtlID)
        {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            default:
                if (BLDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

default:
    break;
}
return bRet;                // No explicit return - return default
}

// *****
//                               Modal Dialog Box: NULLPTR
// *****

// Startup procedure for modal dialog box
int BLD_NULLPtrDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_NULLPtrDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"NULLPTR"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue==-1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"NULLPTR"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_NULLPtrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

```

```
if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
    return bRet;

switch(message)
{

case WM_INITDIALOG:
    bRet = TRUE;    // Default return for WM_INITDIALOG is TRUE
    break;

case WM_COMMAND:
    {
        HWND          hCtrl;

        // Extracting data from message
        hCtrl = (HWND)(UINT)lParam;
        if(!hCtrl)    // Menu input or CR
        {
            if (BLDMenuCommand(hDlg,message,wParam,lParam))
                return TRUE;
        }
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        switch(lpDrawItem->CtlID)
        {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            default:
                if (BLDDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

default:
    break;
}

return bRet;    // No explicit return - return default
}

// *****
//          Modal Dialog Box: DATAERR
// *****

// Startup procedure for modal dialog box
int BLD_DataErrDlgFuncDef(HWND hWnd,char *szDlgName)
```

```
{
DLGPROC      lpProc;
int          ReturnValue;

lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_DataErrDlgProc,hInst);
ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"DATAERR"),
                      hWnd,lpProc);
FreeProcInstance((FARPROC)lpProc);
if (ReturnValue== -1)
    BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"DATAERR"),
                      MB_OK | MB_ICONHAND);
return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_DataErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL      bRet;

    bRet      = FALSE;    // Default return value if not processed

    if(BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND      hCtrl;

            // Extracting data from message
            hCtrl      = (HWND)(UINT)lParam;
            if(!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand(hDlg,message,wParam,lParam))
                    return TRUE;
            }
        }
        break;

    case WM_DRAWITEM:
        {
            LPDRAWITEMSTRUCT lpDrawItem;

            lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
            switch(lpDrawItem->CtlID)
            {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
            }
        }
    }
}
```



```
        break;
    default:
        if (BLDDDrawItem(hDlg, lpDrawItem))
            return TRUE;
        break;
    }
    break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//           Modal Dialog Box: DETAILTXT
// *****

// Startup procedure for modal dialog box
int BLD_DFileErrDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC    lpProc;
    int         ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_DFileErrDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"DETAILTXT"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"DETAILTXT"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_DFileErrDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {
    case WM_INITDIALOG:
        bRet        = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND        hCtrl;

            // Extracting data from message
            hCtrl        = (HWND)(UINT)lParam;
            if (!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand(hDlg, message, wParam, lParam))
                    return TRUE;
            }
        }
    }
}
```

```

    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        switch(lpDrawItem->CtlID)
        {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            default:
                if (BLDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

default:
    break;
}

return bRet;                // No explicit return - return default
}

// *****
//          Modal Dialog Box: NOMORE
// *****

// Startup procedure for modal dialog box
int BLD_NoMoreMatchesDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_NoMoreMatchesDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"NOMORE"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue==-1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"NOMORE"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_NoMoreMatchesDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL          bRet;

```

```
bRet          = FALSE;    // Default return value if not processed

if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
    return bRet;

switch(message)
{

case WM_INITDIALOG:
    bRet      = TRUE;      // Default return for WM_INITDIALOG is TRUE
    BLDInitCtrlFont(hDlg,107,-19,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    break;

case WM_COMMAND:
    {
        HWND          hCtrl;

        // Extracting data from message
        hCtrl          = (HWND)(UINT)lParam;
        if(!hCtrl)      // Menu input or CR
        {
            if (BLDMenuCommand(hDlg,message,wParam,lParam))
                return TRUE;
        }
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        switch(lpDrawItem->CtlID)
        {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"NOTE");
                return TRUE;
                break;
            default:
                if (BLDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

case WM_DESTROY:
    BLDExitCtrlFont(hDlg,107);
    break;

#ifdef WIN32
    case WM_CTLCOLORMSGBOX:
    case WM_CTLCOLOREDIT:
    case WM_CTLCOLORLISTBOX:
    case WM_CTLCOLORBTN:
    case WM_CTLCOLORDLG:
    case WM_CTLCOLORSCROLLBAR:
    case WM_CTLCOLORSTATIC:
#else
    case WM_CTLCOLOR:
#endif
    // Extracting data from message
```



```

        {
            HWND          hCtrl;

#ifdef WIN32
            hCtrl          = (HWND)lParam;
#else
            hCtrl          = (HWND)LOWORD(lParam);
#endif

            switch(GetDlgCtrlID(hCtrl))
            {
                case 107:
                    SetTextColor((HDC)wParam, RGB(0,0,128));
                    SetBkMode((HDC)wParam, OPAQUE);
                    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
                    break;
            }
            break;

        default:
            break;
        }
        return bRet;                // No explicit return - return default
    }

// *****
//                               Modal Dialog Box: MRNERR
// *****

// Startup procedure for modal dialog box
int BLD_MRNErrDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC          lpProc;
    int              ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_MRNErrDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"MRNERR"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"MRNERR"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_MRNErrDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL              bRet;

    bRet              = FALSE;      // Default return value if not processed

    if(BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {

        case WM_INITDIALOG:
            bRet        = TRUE;      // Default return for WM_INITDIALOG is TRUE
            break;
    }

```

```

case WM_COMMAND:
{
    HWND          hCtrl;

    // Extracting data from message
    hCtrl          = (HWND) (UINT) lParam;
    if (!hCtrl)    // Menu input or CR
    {
        if (BLDMenuCommand(hDlg, message, wParam, lParam))
            return TRUE;
    }
}
break;

case WM_DRAWITEM:
{
    LPDRAWITEMSTRUCT lpDrawItem;

    lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
    switch (lpDrawItem->CtlID)
    {
        case 103:
            if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                if (lpDrawItem->CtlType==ODT_BUTTON)
                    BLDDrawIcon(lpDrawItem, "NOTE");
            return TRUE;
            break;
        default:
            if (BLDDrawItem(hDlg, lpDrawItem))
                return TRUE;
            break;
    }
}
break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//          Modal Dialog Box: NOMATCH
// *****

// Startup procedure for modal dialog box
int BLD_MatchErrDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC) MakeProcInstance((FARPROC) BLD_MatchErrDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR) (szDlgName?szDlgName:"NOMATCH"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC) lpProc);
    if (ReturnValue==-1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"NOMATCH"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure

```

```
BOOL BLD_MatchErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL      bRet;

    bRet      = FALSE;    // Default return value if not processed

    if(BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND      hCtrl;

            // Extracting data from message
            hCtrl      = (HWND)(UINT)lParam;
            if(!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand(hDlg,message,wParam,lParam))
                    return TRUE;
            }
        }
        break;

    case WM_DRAWITEM:
        {
            LPDRAWITEMSTRUCT lpDrawItem;

            lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
            switch(lpDrawItem->CtlID)
            {
                case 103:
                    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                        if (lpDrawItem->CtlType==ODT_BUTTON)
                            BLDDDrawIcon(lpDrawItem,"STOP");
                    return TRUE;
                    break;
                case 105:
                    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                        if (lpDrawItem->CtlType==ODT_BUTTON)
                            BLDDDrawIcon(lpDrawItem,"STOP");
                    return TRUE;
                    break;
                default:
                    if(BLDDDrawItem(hDlg,lpDrawItem))
                        return TRUE;
                    break;
            }
        }
        break;

    default:
        break;
    }
    return bRet;    // No explicit return - return default
}
```



```
// *****
//                               Modal Dialog Box: EINORSEQ
// *****

// Startup procedure for modal dialog box
int BLD_EINorSeqErrDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_EINorSeqErrDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"EINORSEQ"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"EINORSEQ"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_EINorSeqErrDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL          bRet;

    bRet          = FALSE;    // Default return value if not processed

    if(BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND          hCtrl;

            // Extracting data from message
            hCtrl          = (HWND)(UINT)lParam;
            if(!hCtrl)      // Menu input or CR
            {
                if (BLDMenuCommand(hDlg, message, wParam, lParam))
                    return TRUE;
            }
        }
        break;

    case WM_DRAWITEM:
        {
            LPDRAWITEMSTRUCT lpDrawItem;

            lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
            switch(lpDrawItem->CtlID)
            {
                case 103:
                    if (lpDrawItem->itemAction == ODA_DRAWENTIRE)
                        if (lpDrawItem->CtlType == ODT_BUTTON)
                            BLDDrawIcon(lpDrawItem, "NOTE");
                    return TRUE;
            }
        }
    }
}
```

```
        break;
    default:
        if (BLDDDrawItem(hDlg,lpDrawItem))
            return TRUE;
        break;
    }
    break;

default:
    break;
}
return bRet;                // No explicit return - return default
}

// *****
//                      Modal Dialog Box: SEQERR
// *****

// Startup procedure for modal dialog box
int BLD_SeqErrDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_SeqErrDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"SEQERR"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"SEQERR"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_SeqErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND        hCtrl;

            // Extracting data from message
            hCtrl        = (HWND)(UINT)lParam;
            if (!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand(hDlg,message,wParam,lParam))
                    return TRUE;
            }
        }
    }
}
```

```
    }
  }
  break;

case WM_DRAWITEM:
  {
    LPDRAWITEMSTRUCT lpDrawItem;

    lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
    switch(lpDrawItem->CtlID)
    {
      case 103:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
          if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawIcon(lpDrawItem,"NOTE");
        return TRUE;
        break;
      default:
        if (BLDDrawItem(hDlg,lpDrawItem))
          return TRUE;
        break;
    }
  }
  break;

default:
  break;
}
return bRet;          // No explicit return - return default
}

// *****
//           Modal Dialog Box: HEADERTXT
// *****

// Startup procedure for modal dialog box
int BLD_HFileDlgFuncDef(HWND hWnd,char *szDlgName)
{
  DLGPROC      lpProc;
  int          ReturnValue;

  lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_HFileDlgProc,hInst);
  ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"HEADERTXT"),
                        hWnd,lpProc);
  FreeProcInstance((FARPROC)lpProc);
  if (ReturnValue==-1)
    BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"HEADERTXT"),
                      MB_OK | MB_ICONHAND);
  return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_HFileDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
  BOOL          bRet;

  bRet          = FALSE;    // Default return value if not processed

  if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
    return bRet;

  switch(message)
```



```
{

case WM_INITDIALOG:
    bRet = TRUE;    // Default return for WM_INITDIALOG is TRUE
    break;

case WM_COMMAND:
    {
        HWND          hCtrl;

        // Extracting data from message
        hCtrl = (HWND) (UINT) lParam;
        if (!hCtrl)    // Menu input or CR
        {
            if (BLDMenuCommand(hDlg,message,wParam,lParam))
                return TRUE;
        }
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
        switch(lpDrawItem->CtlID)
        {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            default:
                if (BLDDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

default:
    break;
}

return bRet;    // No explicit return - return default
}

// *****
//                      Modal Dialog Box: USERERR
// *****

// Startup procedure for modal dialog box
int BLD_UserNumErrDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;
```

```
lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_UserNumErrDlgProc,hInst);
ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"USERERR"),
                      hWnd,lpProc);
FreeProcInstance((FARPROC)lpProc);
if (ReturnValue==-1)
    BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"USERERR"),
                      MB_OK | MB_ICONHAND);
return ReturnValue;
}
```

// Default dialog box procedure

```
BOOL BLD_UserNumErrDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL      bRet;

    bRet      = FALSE;    // Default return value if not processed

    if(BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND      hCtrl;

            // Extracting data from message
            hCtrl      = (HWND)(UINT)lParam;
            if(!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand(hDlg,message,wParam,lParam))
                    return TRUE;
            }
        }
        break;

    case WM_DRAWITEM:
        {
            LPDRAWITEMSTRUCT lpDrawItem;

            lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
            switch(lpDrawItem->CtlID)
            {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            default:
                if(BLDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
            }
        }
    }
}
```

```

        break;
    }
    break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//          Modal Dialog Box: STATOPEN
// *****

// Startup procedure for modal dialog box
int BLD_StatOpenDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_StatOpenDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"STATOPEN"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"STATOPEN"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_StatOpenDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {
        case WM_INITDIALOG:
            bRet        = TRUE;    // Default return for WM_INITDIALOG is TRUE
            break;

        case WM_COMMAND:
            {
                HWND        hCtrl;

                // Extracting data from message
                hCtrl        = (HWND)(UINT)lParam;
                if (!hCtrl)    // Menu input or CR
                {
                    if (BLDMenuCommand(hDlg, message, wParam, lParam))
                        return TRUE;
                }
            }
            break;
    }
}

```



```

case WM_DRAWITEM:
{
    LPDRAWITEMSTRUCT lpDrawItem;

    lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
    switch(lpDrawItem->CtlID)
    {
        case 103:
            if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                if (lpDrawItem->CtlType==ODT_BUTTON)
                    BLDDrawIcon(lpDrawItem,"STOP");
            return TRUE;
            break;
        case 105:
            if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                if (lpDrawItem->CtlType==ODT_BUTTON)
                    BLDDrawIcon(lpDrawItem,"STOP");
            return TRUE;
            break;
        default:
            if (BLDDrawItem(hDlg,lpDrawItem))
                return TRUE;
            break;
    }
}
break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//          Modal Dialog Box: STATEMPTY
// *****

// Startup procedure for modal dialog box
int BLD_StatEmptyDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_StatEmptyDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"STATEMPTY"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue==-1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"STATEMPTY"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_StatEmptyDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;
}

```

```

switch(message)
{

case WM_INITDIALOG:
    bRet      = TRUE;      // Default return for WM_INITDIALOG is TRUE
    break;

case WM_COMMAND:
    {
        HWND      hCtrl;

        // Extracting data from message
        hCtrl      = (HWND) (UINT) lParam;
        if(!hCtrl)      // Menu input or CR
        {
            if (BLDMenuCommand(hDlg,message,wParam,lParam))
                return TRUE;
        }
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
        switch(lpDrawItem->CtlID)
        {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"STOP");
                return TRUE;
                break;
            default:
                if (BLDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

default:
    break;
}

return bRet;      // No explicit return - return default
}

// *****
//          Modal Dialog Box: STATWOPEN
// *****

// Startup procedure for modal dialog box
int BLD_StatWOpenDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC      lpProc;

```

```
int          ReturnValue;

lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_StatWOpenDlgProc,hInst);
ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"STATWOPEN"),
                      hWnd,lpProc);
FreeProcInstance((FARPROC)lpProc);
if (ReturnValue==-1)
    BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"STATWOPEN"),
                      MB_OK | MB_ICONHAND);
return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_StatWOpenDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL      bRet;

    bRet      = FALSE;    // Default return value if not processed

    if(BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND      hCtrl;

            // Extracting data from message
            hCtrl      = (HWND)(UINT)lParam;
            if(!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand(hDlg,message,wParam,lParam))
                    return TRUE;
            }
        }
        break;

    case WM_DRAWITEM:
        {
            LPDRAWITEMSTRUCT lpDrawItem;

            lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
            switch(lpDrawItem->CtlID)
            {
                case 103:
                    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                        if (lpDrawItem->CtlType==ODT_BUTTON)
                            BLDDrawIcon(lpDrawItem,"STOP");
                    return TRUE;
                    break;
                case 105:
                    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                        if (lpDrawItem->CtlType==ODT_BUTTON)
                            BLDDrawIcon(lpDrawItem,"STOP");
                    return TRUE;
                    break;
                default:

```



```
        if (BLDDDrawItem(hDlg, lpDrawItem))
            return TRUE;
        break;
    }
    break;

default:
    break;
}
return bRet;                // No explicit return - return default
}

// *****
//          Modal Dialog Box: STATWRITE
// *****

// Startup procedure for modal dialog box
int BLD_StatWriteErrDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_StatWriteErrDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"STATWRITE"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"STATWRITE"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_StatWriteErrDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {
        case WM_INITDIALOG:
            bRet        = TRUE;    // Default return for WM_INITDIALOG is TRUE
            break;

        case WM_COMMAND:
            {
                HWND        hCtrl;

                // Extracting data from message
                hCtrl        = (HWND)(UINT)lParam;
                if (!hCtrl)    // Menu input or CR
                {
                    if (BLDMenuCommand(hDlg, message, wParam, lParam))
                        return TRUE;
                }
            }
    }
}
```

```
        break;

    case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        switch(lpDrawItem->CtlID)
        {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDDrawIcon(lpDrawItem, "STOP");
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDDrawIcon(lpDrawItem, "STOP");
                return TRUE;
                break;
            default:
                if (BLDDDrawItem(hDlg, lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

default:
    break;
}

return bRet;                // No explicit return - return default
}
```

```
// Filename: PRINT.WMC
// "EAMAT42" Generated by WindowsMAKER Professional.
// Author: Laura L. Downey

//
// *****
// Do not add code here. Add code in the .C file.
//
// This file is maintained by WindowsMAKER Professional.
// As you make changes in your application using WindowsMAKER Professional,
// this file is automatically updated, therefore you never modify this file.
//
//
// For more information,
// see the section "How code is generated" in the documentation.
//
// *****

// *****
// Modal Dialog Box: PBLANKET
// *****

// Startup procedure for modal dialog box
int BLD_PrintBlanketDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_PrintBlanketDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"PBLANKET"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"PBLANKET"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_PrintBlanketDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {
        case WM_INITDIALOG:
            bRet        = TRUE;    // Default return for WM_INITDIALOG is TRUE
            break;

        case WM_COMMAND:
            {
                WORD        wId;
                WORD        notification;
            }
    }
}
```



```
    HWND          hCtrl;

    // Extracting data from message
    wId            = LOWORD(wParam);
    hCtrl          = (HWND) (UINT) lParam;
#ifdef WIN32
    notification = HIWORD(wParam);
#else
    notification = HIWORD(lParam);
#endif
    if(!hCtrl)          // Menu input or CR
    {
        if (BLDMenuCommand(hDlg,message,wParam,lParam))
            return TRUE;
    }
    switch(wId)
    {
    case IDCNT:
        switch(notification)
        {
            case BN_CLICKED:
                EndDialog(hDlg, IDCNT);
                return TRUE;
                break;
            default:
                break;
        }
        break;
    default:
        break;
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
        switch(lpDrawItem->CtlID)
        {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"CASTLE",TRUE);
                return TRUE;
                break;
            case 104:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"ARCADE",TRUE);
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"ARCADE",TRUE);
                return TRUE;
                break;
            case 106:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"CASTLE",TRUE);
                return TRUE;
        }
    }
```

```
        break;
case 107:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"CASTLE",TRUE);
    return TRUE;
    break;
case 108:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"CASTLE",TRUE);
    return TRUE;
    break;
case 109:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"CASTLE",TRUE);
    return TRUE;
    break;
case 110:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"CASTLE",TRUE);
    return TRUE;
    break;
case 111:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"CASTLE",TRUE);
    return TRUE;
    break;
case 112:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"CASTLE",TRUE);
    return TRUE;
    break;
case 113:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"ARCADE",TRUE);
    return TRUE;
    break;
case 114:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"ARCADE",TRUE);
    return TRUE;
    break;
case 115:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"CASTLE",TRUE);
    return TRUE;
    break;
case 116:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"CASTLE",TRUE);
    return TRUE;
    break;
case 117:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
```

```
        BLDDrawBitmap(lpDrawItem, "CASTLE", TRUE);
    return TRUE;
    break;
case 118:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "CASTLE", TRUE);
    return TRUE;
    break;
case 119:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "CASTLE", TRUE);
    return TRUE;
    break;
case 120:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "CASTLE", TRUE);
    return TRUE;
    break;
case 121:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
    break;
case 122:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "CASTLE", TRUE);
    return TRUE;
    break;
case 123:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
    break;
case 124:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "CASTLE", TRUE);
    return TRUE;
    break;
case 125:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
    break;
case 126:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
    break;
case 127:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
    break;
case 128:
```



```

        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDDrawBitmap(lpDrawItem,"ARCADE",TRUE);
        return TRUE;
        break;
    case 129:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDDrawBitmap(lpDrawItem,"ARCADE",TRUE);
        return TRUE;
        break;
    case 130:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDDrawBitmap(lpDrawItem,"ARCADE",TRUE);
        return TRUE;
        break;
    default:
        if (BLDDDrawItem(hDlg,lpDrawItem))
            return TRUE;
        break;
    }
    break;

default:
    break;
}
return bRet;                // No explicit return - return default
}

// *****
//          Modal Dialog Box: PEDETAIL
// *****

// Startup procedure for modal dialog box
int BLD_PrintEmpDetailDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_PrintEmpDetailDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"PEDETAIL"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue==-1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"PEDETAIL"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_PrintEmpDetailDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)

```

```
{

case WM_INITDIALOG:
    bRet      = TRUE;      // Default return for WM_INITDIALOG is TRUE
    break;

case WM_COMMAND:
    {
        HWND      hCtrl;

        // Extracting data from message
        hCtrl      = (HWND) (UINT) lParam;
        if (!hCtrl)      // Menu input or CR
        {
            if (BLDMenuCommand(hDlg, message, wParam, lParam))
                return TRUE;
        }
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
        switch(lpDrawItem->CtlID)
        {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
                return TRUE;
                break;
            case 104:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
                return TRUE;
                break;
            case 106:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
                return TRUE;
                break;
            case 107:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
                return TRUE;
                break;
            case 108:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
                return TRUE;
                break;
            case 109:
```

```
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
break;
case 110:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
break;
case 111:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
break;
case 112:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
break;
case 113:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
break;
case 114:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
break;
case 115:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
break;
case 116:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
break;
case 117:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
break;
case 118:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
break;
case 119:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem, "ARCADE", TRUE);
    return TRUE;
```



```
        break;
    case 120:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDDrawBitmap(lpDrawItem,"ARCADE",TRUE);
        return TRUE;
        break;
    default:
        if (BLDDDrawItem(hDlg,lpDrawItem))
            return TRUE;
        break;
    }
    break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//          Modal Dialog Box: PHDETAIL
// *****

// Startup procedure for modal dialog box
int BLD_PrintHeaderDetailDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_PrintHeaderDetailDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"PHDETAIL"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"PHDETAIL"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_PrintHeaderDetailDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            HWND        hCtrl;
```

```
// Extracting data from message
hCtrl      = (HWND) (UINT) lParam;
if (!hCtrl)      // Menu input or CR
{
    if (BLDMenuCommand(hDlg, message, wParam, lParam))
        return TRUE;
}
break;

case WM_DRAWITEM:
{
    LPDRAWITEMSTRUCT lpDrawItem;

    lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
    switch (lpDrawItem->CtlID)
    {
        case 103:
            if (lpDrawItem->itemAction == ODA_DRAWENTIRE)
                if (lpDrawItem->CtlType == ODT_BUTTON)
                    BLDDrawBitmap(lpDrawItem, "RIVETS", TRUE);
            return TRUE;
            break;
        case 104:
            if (lpDrawItem->itemAction == ODA_DRAWENTIRE)
                if (lpDrawItem->CtlType == ODT_BUTTON)
                    BLDDrawBitmap(lpDrawItem, "RIVETS", TRUE);
            return TRUE;
            break;
        case 105:
            if (lpDrawItem->itemAction == ODA_DRAWENTIRE)
                if (lpDrawItem->CtlType == ODT_BUTTON)
                    BLDDrawBitmap(lpDrawItem, "RIVETS", TRUE);
            return TRUE;
            break;
        case 106:
            if (lpDrawItem->itemAction == ODA_DRAWENTIRE)
                if (lpDrawItem->CtlType == ODT_BUTTON)
                    BLDDrawBitmap(lpDrawItem, "RIVETS", TRUE);
            return TRUE;
            break;
        case 107:
            if (lpDrawItem->itemAction == ODA_DRAWENTIRE)
                if (lpDrawItem->CtlType == ODT_BUTTON)
                    BLDDrawBitmap(lpDrawItem, "RIVETS", TRUE);
            return TRUE;
            break;
        case 108:
            if (lpDrawItem->itemAction == ODA_DRAWENTIRE)
                if (lpDrawItem->CtlType == ODT_BUTTON)
                    BLDDrawBitmap(lpDrawItem, "RIVETS", TRUE);
            return TRUE;
            break;
        case 109:
            if (lpDrawItem->itemAction == ODA_DRAWENTIRE)
                if (lpDrawItem->CtlType == ODT_BUTTON)
                    BLDDrawBitmap(lpDrawItem, "RIVETS", TRUE);
            return TRUE;
            break;
        case 110:
            if (lpDrawItem->itemAction == ODA_DRAWENTIRE)
                if (lpDrawItem->CtlType == ODT_BUTTON)
                    BLDDrawBitmap(lpDrawItem, "RIVETS", TRUE);
            return TRUE;
    }
}
```

```
        break;
case 111:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"RIVETS",TRUE);
    return TRUE;
    break;
case 112:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"RIVETS",TRUE);
    return TRUE;
    break;
case 113:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"RIVETS",TRUE);
    return TRUE;
    break;
case 114:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"RIVETS",TRUE);
    return TRUE;
    break;
case 115:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"RIVETS",TRUE);
    return TRUE;
    break;
case 116:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"RIVETS",TRUE);
    return TRUE;
    break;
case 117:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"RIVETS",TRUE);
    return TRUE;
    break;
case 118:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"RIVETS",TRUE);
    return TRUE;
    break;
case 119:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"RIVETS",TRUE);
    return TRUE;
    break;
case 120:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"RIVETS",TRUE);
    return TRUE;
    break;
default:
    if (BLDDrawItem(hDlg,lpDrawItem))
        return TRUE;
```



```

        break;
    }
    break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//          Modal Dialog Box: PINFO
// *****

// Startup procedure for modal dialog box
int BLD_Function5DlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC    lpProc;
    int         ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_Function5DlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"PINFO"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"PINFO"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_Function5DlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch (message)
    {

    case WM_INITDIALOG:
        bRet        = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            WORD        wId;
            WORD        notification;
            HWND        hCtrl;

            // Extracting data from message
            wId        = LOWORD(wParam);
            hCtrl       = (HWND)(UINT)lParam;

#ifdef WIN32
            notification = HIWORD(wParam);
#else
            notification = HIWORD(lParam);
#endif
        }
    }
}

```

```
    if(!hCtrl)          // Menu input or CR
    {
        if (BLDMenuCommand(hDlg,message,wParam,lParam))
            return TRUE;
    }
    switch(wId)
    {
    case 101:
        switch(notification)
        {
            case BN_CLICKED:
                EndDialog(hDlg,101);
                return TRUE;
                break;
            default:
                break;
        }
        break;
    default:
        break;
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        switch(lpDrawItem->CtlID)
        {
            case 103:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"ARGYLE",TRUE);
                return TRUE;
                break;
            case 104:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"ARGYLE",TRUE);
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"ARGYLE",TRUE);
                return TRUE;
                break;
            case 106:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"ARGYLE",TRUE);
                return TRUE;
                break;
            case 107:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"ARGYLE",TRUE);
                return TRUE;
                break;
            case 108:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
```

```
        BLDDDrawBitmap(lpDrawItem, "ARGYLE", TRUE);
    return TRUE;
    break;
case 109:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDDrawBitmap(lpDrawItem, "ARGYLE", TRUE);
    return TRUE;
    break;
case 110:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDDrawBitmap(lpDrawItem, "ARGYLE", TRUE);
    return TRUE;
    break;
case 111:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDDrawBitmap(lpDrawItem, "ARGYLE", TRUE);
    return TRUE;
    break;
case 112:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDDrawBitmap(lpDrawItem, "ARGYLE", TRUE);
    return TRUE;
    break;
case 113:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDDrawBitmap(lpDrawItem, "ARGYLE", TRUE);
    return TRUE;
    break;
case 114:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDDrawBitmap(lpDrawItem, "ARGYLE", TRUE);
    return TRUE;
    break;
case 115:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDDrawBitmap(lpDrawItem, "ARGYLE", TRUE);
    return TRUE;
    break;
case 116:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDDrawBitmap(lpDrawItem, "ARGYLE", TRUE);
    return TRUE;
    break;
case 117:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDDrawBitmap(lpDrawItem, "ARGYLE", TRUE);
    return TRUE;
    break;
case 118:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDDrawBitmap(lpDrawItem, "ARGYLE", TRUE);
    return TRUE;
    break;
case 119:
```



```

        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDrawBitmap(lpDrawItem,"ARGYLE",TRUE);
        return TRUE;
        break;
    case 120:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDrawBitmap(lpDrawItem,"ARGYLE",TRUE);
        return TRUE;
        break;
    default:
        if (BLDDrawItem(hDlg,lpDrawItem))
            return TRUE;
        break;
    }
    break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//          Modal Dialog Box: PTOTALS
// *****

// Startup procedure for modal dialog box
int BLD_TotNowPrintDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_TotNowPrintDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"PTOTALS"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"PTOTALS"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_TotNowPrintDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet        = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

```

```
case WM_COMMAND:
{
    WORD          wId;
    WORD          notification;
    HWND          hCtrl;

    // Extracting data from message
    wId           = LOWORD(wParam);
    hCtrl         = (HWND) (UINT) lParam;
#ifdef WIN32
    notification = HIWORD(wParam);
#else
    notification = HIWORD(lParam);
#endif
    if(!hCtrl)           // Menu input or CR
    {
        if (BLDMenuCommand(hDlg,message,wParam,lParam))
            return TRUE;
    }
    switch(wId)
    {
    case 101:
        switch(notification)
        {
            case BN_CLICKED:
                EndDialog(hDlg,101);
                return TRUE;
                break;
            default:
                break;
        }
        break;
    default:
        break;
    }
    break;

case WM_DRAWITEM:
{
    LPDRAWITEMSTRUCT lpDrawItem;

    lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
    switch(lpDrawItem->CtlID)
    {
    case 103:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
        return TRUE;
        break;
    case 104:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
        return TRUE;
        break;
    case 105:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
        return TRUE;
        break;
    case 106:
```

```
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
    return TRUE;
break;
case 107:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
    return TRUE;
break;
case 108:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
    return TRUE;
break;
case 109:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
    return TRUE;
break;
case 110:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
    return TRUE;
break;
case 111:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
    return TRUE;
break;
case 112:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
    return TRUE;
break;
case 113:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
    return TRUE;
break;
case 115:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
    return TRUE;
break;
case 116:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
    return TRUE;
break;
case 117:
    if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
        if (lpDrawItem->CtlType==ODT_BUTTON)
            BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
    return TRUE;
```



```

        break;
    case 118:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
        return TRUE;
        break;
    case 119:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
        return TRUE;
        break;
    case 120:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
        return TRUE;
        break;
    case 121:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDrawBitmap(lpDrawItem,"ZIGZAG",TRUE);
        return TRUE;
        break;
    default:
        if (BLDDrawItem(hDlg,lpDrawItem))
            return TRUE;
        break;
    }
    break;

default:
    break;
}

return bRet;          // No explicit return - return default
}

// *****
//           Modal Dialog Box: NCOPY
// *****

// Startup procedure for modal dialog box
int BLD_GetNumCopyDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_GetNumCopyDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"NCOPY"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"NCOPY"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_GetNumCopyDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{

```

```
    BOOL          bRet;

    bRet          = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;      // Default return for WM_INITDIALOG is TRUE
        BLDInitSolidBrush(hDlg,RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg,103),RGB(192,192,192));
        break;

    case WM_COMMAND:
        {
            HWND          hCtrl;

            // Extracting data from message
            hCtrl          = (HWND) (UINT) lParam;
            if(!hCtrl)      // Menu input or CR
            {
                if (BLDMenuCommand(hDlg,message,wParam,lParam))
                    return TRUE;
            }
        }
        break;

    case WM_DRAWITEM:
        {
            LPDRAWITEMSTRUCT lpDrawItem;

            lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
            if (BLDDrawItem(hDlg,lpDrawItem))
                return TRUE;
        }
        break;
        break;

    case WM_DESTROY:
        BLDExitBrush(hDlg);
        BLDExitBrush(GetDlgItem(hDlg,103));
        break;

#ifdef WIN32
    case WM_CTLCOLOORMSGBOX:
    case WM_CTLCOLOREDIT:
    case WM_CTLCOLORLISTBOX:
    case WM_CTLCOLORBTN:
    case WM_CTLCOLORDLG:
    case WM_CTLCOLORSCROLLBAR:
    case WM_CTLCOLORSTATIC:
#else
    case WM_CTLCOLOR:
#endif
        // Extracting data from message
        {
            HWND          hCtrl;

#ifdef WIN32
            hCtrl          = (HWND) lParam;
        }
    }
}
```

```
        hCtrl          = (HWND) LOWORD(lParam);
#endif
#ifdef WIN32
        if(message == WM_CTLCOLORDLG)
            return (BOOL) BLDctlColorBrushSetOrg(hDlg, (HDC) wParam);
#else
        if(HIWORD(lParam) == CTLCOLOR_DLG)
            return (BOOL) BLDctlColorBrushSetOrg(hDlg, (HDC) wParam);
#endif
        switch(GetDlgCtrlID(hCtrl))
        {
        case 103:
            SetBkMode((HDC) wParam, TRANSPARENT);
            return (BOOL) BLDctlColorPropBrush(hCtrl);
            break;
        }
        break;

default:
    break;
}
return bRet;          // No explicit return - return default
}
```



```
// Filename: SERVICE.WMC
// "EAMAT42" Generated by WindowsMAKER Professional.
// Author: Laura L. Downey

//
// *****
// Do not add code here. Add code in the .C file.
//
// This file is maintained by WindowsMAKER Professional.
// As you make changes in your application using WindowsMAKER Professional,
// this file is automatically updated, therefore you never modify this file.
//
//
// For more information,
// see the section "How code is generated" in the documentation.
//
// *****
//
```

```
// *****
// ERROR MESSAGE HANDLING
// *****
```

```
int BLDDisplayMessageDef(HWND hWnd,UINT uMsg,char *pContext,int iType)
{
    int i, j;
    char szMessage[200+1];

    if (uMsg)
    {
        if (!LoadString(hInst,uMsg,szMessage,200))
        {
            MessageBox(hWnd,BLDLOADERROR,BLDMAINCAPTION,
                MB_OK|MB_SYSTEMMODAL|MB_ICONHAND);
            return FALSE;
        }
    }
    else
        szMessage[0]=0;

    if (pContext)
    {
        i = lstrlen(szMessage);
        j = lstrlen(pContext);
        if (i + j + 1 <= 200)
        {
            lstrcat(szMessage, " ");
            lstrcat(szMessage, pContext);
        }
    }

    return MessageBox(hWnd,szMessage,BLDMAINCAPTION,iType);
}
```

```
// *****
// FUNCTIONS FOR DRAWING GRAPHIC BUTTONS
// *****
```

```
BOOL BLDBitmapToScreenDef(HDC hDestDC, char *pBitmapName,
```

```
int X,int Y,int nWidth,int nHeight,
DWORD dwRop,BOOL bStretch)
{
    HDC          hMemDC;
    BITMAP       Bitmap;
    HBITMAP      hBitmap;

    if (!hBMPInst)
        hBMPInst = hInst;

    hBitmap = BLDLoadBitmap(hBMPInst,pBitmapName);

    if (!hBitmap)
    {
        BLDDisplayMessage(GetActiveWindow(),BLD_CannotLoadBitmap,pBitmapName,
                           MB_OK | MB_ICONASTERISK);
        return FALSE;
    }

    hMemDC = CreateCompatibleDC(hDestDC);

    if (!hMemDC)
    {
        DeleteObject(hBitmap);
        return FALSE;
    }
    if (!SelectObject(hMemDC,hBitmap))
    {
        DeleteObject(hBitmap);
        DeleteDC(hMemDC);
        return FALSE;
    }

    if (bStretch)
    {
        if (!GetObject(hBitmap,sizeof(BITMAP),(LPSTR)&Bitmap))
        {
            DeleteObject(hBitmap);
            return FALSE;
        }

        StretchBlt(hDestDC,
                    X,
                    Y,
                    nWidth,
                    nHeight,
                    hMemDC,
                    0,
                    0,
                    Bitmap.bmWidth,
                    Bitmap.bmHeight,
                    dwRop);
    }
    else
    {
        BitBlt(hDestDC,X,Y,nWidth,nHeight,hMemDC,0,0,dwRop);
    }

    DeleteDC(hMemDC);
    DeleteObject(hBitmap);
    return TRUE;
}
```

```
BOOL BLDDDrawIconDef(LPDRAWITEMSTRUCT lpDrawItem,char *pIconName)
{
    HICON      hIcon;

    hIcon = LoadIcon(hInst,pIconName);
    if (!hIcon)
    {
        BLDDisplayMessage(GetActiveWindow(),BLD_CannotLoadIcon,pIconName,
            MB_OK | MB_ICONASTERISK);
        return FALSE;
    }

    SetMapMode(lpDrawItem->hDC,MM_TEXT);
    return DrawIcon(lpDrawItem->hDC,0,0,hIcon);
}

BOOL BLDDDrawBitmapDef(LPDRAWITEMSTRUCT lpDrawItem,char *pBitmapName,BOOL bStretch)
{
    int      iRaster;

    iRaster = GetDeviceCaps(lpDrawItem->hDC,RASTERCAPS);
    if ((iRaster&RC_BITBLT)!=RC_BITBLT)
        return FALSE; // Device cannot display bitmap

    return BLDBitmapToScreen(lpDrawItem->hDC,pBitmapName,
        lpDrawItem->rcItem.left,
        lpDrawItem->rcItem.top,
        lpDrawItem->rcItem.right-lpDrawItem->rcItem.left,
        lpDrawItem->rcItem.bottom-lpDrawItem->rcItem.top,
        SRCCOPY,bStretch);
}

BOOL BLDDDrawBkgndIconDef(HWND hDlg,PAINTSTRUCT *pPs,char *pIconName,int iCtrlID)
{
    RECT      Rect,Dummy;
    HWND      CtrlhWnd;
    HICON      hIcon;

    CtrlhWnd=GetDlgItem(hDlg,iCtrlID);
    if(!CtrlhWnd)
        return FALSE;
    GetWindowRect(CtrlhWnd, &Rect);
    ScreenToClient(hDlg, (LPPOINT)&Rect.left);
    ScreenToClient(hDlg, (LPPOINT)&Rect.right);

    hIcon = LoadIcon(hInst,pIconName);
    if (!hIcon)
    {
        BLDDisplayMessage(GetActiveWindow(),BLD_CannotLoadIcon,pIconName,
            MB_OK | MB_ICONASTERISK);
        return FALSE;
    }

    if(!IntersectRect(&Dummy, &Rect, &pPs->rcPaint))
        return TRUE; // No intersection

    SetMapMode(pPs->hdc,MM_TEXT);
    return DrawIcon(pPs->hdc,Rect.left,Rect.top,hIcon);
}
```



```
BOOL BLDDrawBkgndBitmapDef(HWND hDlg, PAINTSTRUCT *pPs, char *pBitmapName,
                           int iCtrlID, BOOL bStretch, BOOL bCtrl)
{
    int          iRaster;
    RECT         Rect, Dummy;
    HWND         CtrlhWnd;
    int          xScrolled, yScrolled;

    iRaster = GetDeviceCaps(pPs->hdc, RASTERCAPS);
    if ((iRaster & RC_BITBLT) != RC_BITBLT)
        return FALSE; // Device cannot display bitmap

    if (bCtrl)
    {
        CtrlhWnd = GetDlgItem(hDlg, iCtrlID);
        if (!CtrlhWnd)
            return FALSE;
        GetWindowRect(CtrlhWnd, &Rect);
        ScreenToClient(hDlg, (LPPOINT)&Rect.left);
        ScreenToClient(hDlg, (LPPOINT)&Rect.right);
    }
    else
    {
        GetClientRect(hDlg, &Rect);
        if (!bStretch)
        {
            BLDGetDlgScrolled(hDlg, &xScrolled, &yScrolled);
            Rect.left -= xScrolled;
            Rect.top  -= yScrolled;
        }
    }

    if (!IntersectRect(&Dummy, &Rect, &pPs->rcPaint))
        return TRUE; // No intersection

    return BLDBitmapToScreen(pPs->hdc, pBitmapName,
                             Rect.left,
                             Rect.top,
                             Rect.right-Rect.left,
                             Rect.bottom-Rect.top,
                             SRCCOPY, bStretch);
}

BOOL BLDDrawAutoStateDef(LPDRAWITEMSTRUCT lpDrawItem, char *szResource, BOOL bBitmap,
                        BOOL bStretch)
{
    int          x, y, dx, dy;
    BOOL         bDown, bActive;
    HBRUSH       hOldBrush;
    int          incr, i, j, bGray;
    COLORREF     color;
    HICON        hIcon;

    if (lpDrawItem->CtlType != ODT_BUTTON)
        return FALSE;

    if ((lpDrawItem->itemAction & (ODA_SELECT | ODA_DRAWENTIRE)) == 0)
        return FALSE;
```

```
bDown = (lpDrawItem->itemState & ODS_SELECTED) != 0;
bActive = (lpDrawItem->itemState & ODS_DISABLED) == 0;

x = lpDrawItem->rcItem.left;
y = lpDrawItem->rcItem.top;
dx = lpDrawItem->rcItem.right-lpDrawItem->rcItem.left-6;
dy = lpDrawItem->rcItem.bottom-lpDrawItem->rcItem.top-6;

incr = bDown ? 4 : 3;

if(!*szResource)
{
    hOldBrush = SelectObject(lpDrawItem->hDC, GetStockObject(LTGRAY_BRUSH));
    PatBlt(lpDrawItem->hDC, x+incr, y+incr, dx, dy, PATCOPY);
    SelectObject(lpDrawItem->hDC, hOldBrush);
}
else
{
    if(bBitmap)
    {
        if(!BLDBitmapToScreen(lpDrawItem->hDC, szResource,
                               x+incr,
                               y+incr,
                               dx,
                               dy,
                               SRCCOPY, bStretch))
            return FALSE;
    }
    else
    {
        if (! (hIcon=LoadIcon(hInst, szResource)))
            return FALSE;
        DrawIcon(lpDrawItem->hDC, x+incr, y+incr, hIcon);
    }
}

// Make the bitmap grayed
if (!bActive)
{
    color = BLD_LTGRAY;

    for (j=y+incr; j<dy; ++j)
    {
        bGray = j % 2;
        for (i=x+incr; i<dx; ++i)
        {
            if (bGray)
                SetPixel(lpDrawItem->hDC, i, j, color);
            bGray = !bGray;
        }
    }
}

return BLDDrawFrame(lpDrawItem->hDC, x, y, dx, dy, bDown);
}
```

```
BOOL BLDDrawStateBitmapDef(LPDRAWITEMSTRUCT lpDrawItem, char *szNormal, char *szFocus,
                           char *szSelected, char *szDisabled, BOOL bStretch)
{
    if( !*szFocus && !*szSelected && !*szDisabled)
    {
        return BLDDrawAutoState(lpDrawItem, szNormal, TRUE, bStretch);
    }
}
```

```
    }
    if((lpDrawItem->itemState & ODS_DISABLED) && *szDisabled)
        return BLDDDrawBitmap(lpDrawItem,szDisabled,bStretch);
    if((lpDrawItem->itemState & ODS_SELECTED) && *szSelected)
        return BLDDDrawBitmap(lpDrawItem,szSelected,bStretch);
    if((lpDrawItem->itemState & ODS_FOCUS) && *szFocus)
        return BLDDDrawBitmap(lpDrawItem,szFocus,bStretch);
    if(*szNormal)
        return BLDDDrawBitmap(lpDrawItem,szNormal,bStretch);
    return TRUE;
}
```

```
BOOL BLDDDrawStateIconDef(LPDRAWITEMSTRUCT lpDrawItem,char *szNormal,char *szFocus,
                          char *szSelected,char *szDisabled)
```

```
{
    if( !*szFocus && !*szSelected && !*szDisabled)
    {
        return BLDDDrawAutoState(lpDrawItem, szNormal, FALSE, FALSE);
    }
    if((lpDrawItem->itemState & ODS_DISABLED) && *szDisabled)
        return BLDDDrawIcon(lpDrawItem,szDisabled);
    if((lpDrawItem->itemState & ODS_SELECTED) && *szSelected)
        return BLDDDrawIcon(lpDrawItem,szSelected);
    if((lpDrawItem->itemState & ODS_FOCUS) && *szFocus)
        return BLDDDrawIcon(lpDrawItem,szFocus);
    if(*szNormal)
        return BLDDDrawIcon(lpDrawItem,szNormal);
    return TRUE;
}
```

```
BOOL BLDDDrawItemDef(HWND hDlg,LPDRAWITEMSTRUCT lpDrawItem)
```

```
{
    char szStr[20];

    if(lpDrawItem->CtlType == ODT_BUTTON)
    {
        GetWindowText(lpDrawItem->hwndItem,szStr,20);
        if(lstrcmpi( (LPSTR)szStr, (LPSTR)"WMP3DBUTTON" ) == 0 )
        {
            BLDDDrawStateBitmap(lpDrawItem,"","","","",TRUE);
            return TRUE;
        }
    }
    return FALSE;
}
```

```
BOOL BLDDDrawFrame(HDC hDC,int x,int y,int dx,int dy,BOOL bDown)
```

```
{
    HPEN      hold,hBlack,hLtGray,hGray,hWhite;

    hBlack   = CreatePen(0,1,BLD_BLACK);
    hGray    = CreatePen(0,1,GetDeviceCaps(hDC, NUMCOLORS) > 2 ? BLD_GRAY : BLD_WHITE);
    hWhite   = CreatePen(0,1,BLD_WHITE);
    hLtGray  = CreatePen(0,1,GetDeviceCaps(hDC, NUMCOLORS) > 2 ? BLD_LTGRAY : BLD_WHITE);

    // Paint the frame
    hold = SelectObject(hDC, hBlack);
    BLDMoveTo(hDC,x+1,y);
    LineTo(hDC, x+dx+5, y);
}
```



```
BLDMoveTo(hDC,x+1,y+dy+5);
LineTo(hDC, x+dx+5, y+dy+5);

BLDMoveTo(hDC,x,y+1);
LineTo(hDC, x, y+dy+5);
BLDMoveTo(hDC,x+dx+5,y+1);
LineTo(hDC, x+dx+5, y+dy+5);

if (bDown)
{
    BLDMoveTo(hDC,x+1,y+1);
    LineTo(hDC, x+dx+5, y+1);
    BLDMoveTo(hDC,x+1,y+2);
    LineTo(hDC, x+1, y+dy+5);
    BLDMoveTo(hDC,x+dx+4,y+dy+4);
    LineTo(hDC, x, y+dy+4);
    BLDMoveTo(hDC,x+dx+4,y+dy+3);
    LineTo(hDC, x+dx+4, y);

    SelectObject(hDC,hGray);
    BLDMoveTo(hDC,x+2,y+2);
    LineTo(hDC,x+dx+4,y+2);
    BLDMoveTo(hDC,x+2,y+3);
    LineTo(hDC,x+2,y+dy+4);

    SelectObject(hDC,hLtGray);
    BLDMoveTo(hDC,x+3,y+3);
    LineTo(hDC,x+dx+4,y+3);
    BLDMoveTo(hDC,x+3,y+4);
    LineTo(hDC,x+3,y+dy+4);
}
else
{
    SelectObject(hDC,hWhite);
    BLDMoveTo(hDC,x+1,y+1);
    LineTo(hDC,x+dx+4,y+1);
    BLDMoveTo(hDC,x+1,y+2);
    LineTo(hDC,x+1,y+dy+4);
    BLDMoveTo(hDC,x+2,y+2);
    LineTo(hDC,x+dx+3,y+2);
    BLDMoveTo(hDC,x+2,y+3);
    LineTo(hDC,x+2,y+dy+3);

    SelectObject(hDC, hGray);
    BLDMoveTo(hDC,x+dx+4,y+dy+4);
    LineTo(hDC,x,y+dy+4);
    BLDMoveTo(hDC,x+dx+4,y+dy+3);
    LineTo(hDC,x+dx+4,y);
    BLDMoveTo(hDC,x+dx+3,y+dy+3);
    LineTo(hDC,x+1,y+dy+3);
    BLDMoveTo(hDC,x+dx+3,y+dy+2);
    LineTo(hDC,x+dx+3,y+1);
}

SelectObject(hDC, hold);
DeleteObject(hBlack);
DeleteObject(hWhite);
DeleteObject(hGray);
DeleteObject(hLtGray);
return TRUE;
}
```

```
static BOOL BLDMoveTo(HDC hDC,int x,int y)
{
#ifdef WIN32
    return MoveToEx(hDC,x,y,NULL);
#else
    return (BOOL)MoveTo(hDC,x,y);
#endif
}

// *****
//          FUNCTIONS FOR DIALOG BOX SCROLLING
// *****

void BLDGetDlgScrolledDef(HWND hDlg,int *pxScrolled,int *pyScrolled)
{
    *pxScrolled=(int)GetProp(hDlg,"BLDPROPHSCROLLED");
    *pyScrolled=(int)GetProp(hDlg,"BLDPROPVSCROLLED");
}

void BLDSetDlgScrolledDef(HWND hDlg,int xScrolled,int yScrolled)
{
    if(xScrolled)
        SetProp(hDlg,"BLDPROPHSCROLLED",(HANDLE)xScrolled);
    else
        if(GetProp(hDlg,"BLDPROPHSCROLLED"))
            RemoveProp(hDlg,"BLDPROPHSCROLLED");

    if(yScrolled)
        SetProp(hDlg,"BLDPROPVSCROLLED",(HANDLE)yScrolled);
    else
        if(GetProp(hDlg,"BLDPROPVSCROLLED"))
            RemoveProp(hDlg,"BLDPROPVSCROLLED");
}

BOOL BLDExitScrollDlgDef(HWND hDlg)
{
    if(GetProp(hDlg,"BLDPROPHSCROLLED"))
        RemoveProp(hDlg,"BLDPROPHSCROLLED");
    if(GetProp(hDlg,"BLDPROPVSCROLLED"))
        RemoveProp(hDlg,"BLDPROPVSCROLLED");
    return TRUE;
}

void BLDFindCtrlsRightBottomDef(HWND hDlg,int *xRight,int *yBottom)
{
    HWND      CtrlhWnd;
    RECT      Rect;

    *xRight=0;
    *yBottom=0;

    CtrlhWnd = GetWindow(hDlg,GW_CHILD);

    while(CtrlhWnd)
    {
        GetWindowRect(CtrlhWnd,&Rect);
        ScreenToClient(hDlg,(LPPOINT)&Rect.right);
        *xRight = max(*xRight,Rect.right);
        *yBottom = max(*yBottom,Rect.bottom);
        CtrlhWnd = GetWindow(CtrlhWnd,GW_HWNDNEXT);
    }
}
```

```
    }  
}
```

```
void BLDCalcScrollRangesDef(HWND hDlg, int *xRange, int *yRange, int xScrolled,  
                           int yScrolled, int iRightOf, int iBelow)
```

```
{  
    RECT      ClientRect;  
    int       xRight, yBottom;  
  
    BLDFindCtrlsRightBottom(hDlg, &xRight, &yBottom);  
    GetClientRect(hDlg, &ClientRect);  
    xRight += xScrolled + iRightOf;  
    yBottom += yScrolled + iBelow;  
  
    *xRange = max(0, xRight - ClientRect.right);  
    *yRange = max(0, yBottom - ClientRect.bottom);  
}
```

```
BOOL BLDScrollDlgDef(HWND hDlg, UINT message, int nSclCode, int nPos, int iVertLine,  
                    int iHorLine, int iVertPage, int iHorPage, int iRightOf,  
                    int iBelow, BOOL bInvalidate)
```

```
{  
    int       xScroll, yScroll;  
    int       xRange, yRange;  
    int       iThumb;  
    BOOL      bDlgUnits;  
    int       xScrolled;  
    int       yScrolled;
```

```
    BLDGetDlgScrolled(hDlg, &xScrolled, &yScrolled);
```

```
    xScroll = yScroll = 0;
```

```
    BLDCalcScrollRanges(hDlg, &xRange, &yRange, xScrolled, yScrolled, iRightOf, iBelow);
```

```
    bDlgUnits = FALSE;
```

```
    switch(message)
```

```
    {
```

```
    case WM_VSCROLL:
```

```
        switch(nSclCode)
```

```
        {
```

```
        case SB_BOTTOM:
```

```
            yScroll = yRange - yScrolled;
```

```
            bDlgUnits = FALSE;
```

```
            break;
```

```
        case SB_ENDSCROLL:
```

```
            break;
```

```
        case SB_LINEDOWN:
```

```
            yScroll = iVertLine;
```

```
            bDlgUnits = TRUE;
```

```
            break;
```

```
        case SB_LINEUP:
```

```
            yScroll = -iVertLine;
```

```
            bDlgUnits = TRUE;
```

```
            break;
```

```
        case SB_PAGEDOWN:
```

```
            if(iVertPage)
```

```
            {
```

```
                RECT Rect;
```

```
                GetClientRect(hDlg, &Rect);
```

```
                yScroll = MulDiv(Rect.bottom, iVertPage, 100);
```



```
        bDlgUnits=FALSE;
    }
    break;
case SB_PAGEUP:
    if(iVertPage)
    {
        RECT Rect;
        GetClientRect(hDlg,&Rect);
        yScroll=-MulDiv(Rect.bottom,iVertPage,100);
        bDlgUnits=FALSE;
    }
    break;
case SB_THUMBPOSITION:
    iThumb =nPos;
    yScroll=-yScrolled + MulDiv(iThumb,yRange,100);
    bDlgUnits=FALSE;
    break;
case SB_THUMBTRACK:    // No Support
    break;
case SB_TOP:
    yScroll=-yScrolled;
    bDlgUnits=FALSE;
    break;
}
break;
case WM_HSCROLL:
    switch(nSclCode)
    {
        case SB_BOTTOM:
            xScroll=xRange-xScrolled;
            bDlgUnits=FALSE;
            break;
        case SB_ENDSCROLL:
            break;
        case SB_LINEDOWN:
            xScroll=iHorLine;
            bDlgUnits=TRUE;
            break;
        case SB_LINEUP:
            xScroll=-iHorLine;
            bDlgUnits=TRUE;
            break;
        case SB_PAGEDOWN:
            if(iVertPage)
            {
                RECT Rect;
                GetClientRect(hDlg,&Rect);
                xScroll=MulDiv(Rect.right,iHorPage,100);
                bDlgUnits=FALSE;
            }
            break;
        case SB_PAGEUP:
            if(iVertPage)
            {
                RECT Rect;
                GetClientRect(hDlg,&Rect);
                xScroll=-MulDiv(Rect.bottom,iHorPage,100);
                bDlgUnits=FALSE;
            }
            break;
        case SB_THUMBPOSITION:
            iThumb =nPos;
            xScroll=-xScrolled + MulDiv(iThumb,xRange,100);
            bDlgUnits=FALSE;
```

```
        break;
    case SB_THUMBTRACK:    // No Support
        break;
    case SB_TOP:
        xScroll=-xScrolled;
        bDlgUnits=FALSE;
        break;
    }
    break;
}

if(xScroll || yScroll)
{
    int x,y;
    int oldx,oldy;

    x=y=0;
    if(bDlgUnits)
    {
        BOOL xNeg,yNeg;
        xNeg=yNeg=FALSE;
        if(xScroll < 0)
        {
            xScroll=-xScroll;
            xNeg=TRUE;
        }
        if(yScroll < 0)
        {
            yScroll=-yScroll;
            yNeg=TRUE;
        }
        xScroll=(xScroll * LOWORD(GetDialogBaseUnits()))/4;
        yScroll=(yScroll * HIWORD(GetDialogBaseUnits()))/8;
        if(xNeg)
            xScroll=-xScroll;
        if(yNeg)
            yScroll=-yScroll;
    }

    oldx=xScrolled;
    oldy=yScrolled;
    xScrolled+=xScroll;
    yScrolled+=yScroll;

    if(xScrolled > xRange)
        xScrolled = xRange;
    if(xScrolled < 0)
        xScrolled=0;

    if(yScrolled > yRange)
        yScrolled = yRange;
    if(yScrolled < 0)
        yScrolled=0;

    xScroll=xScrolled - oldx;
    yScroll=yScrolled - oldy;

    if(xScroll || yScroll)
    {
        ScrollWindow(hDlg,-xScroll,-yScroll,NULL,NULL);
        if(xRange)
        {
            x=MulDiv(xScrolled,100,xRange);
            SetScrollPos(hDlg,SB_HORZ,x,TRUE);
        }
    }
}
```

```
    }
    if (yRange)
    {
        y=MulDiv(yScrolled,100,yRange);
        SetScrollPos(hDlg,SB_VERT,y,TRUE);
    }
    if(bInvalidate)
        InvalidateRect(hDlg,NULL,FALSE);
    BLDSetDlgScrolled(hDlg,xScrolled,yScrolled);
}

return TRUE;
}

// *****
//      FUNCTION FOR CREATING CONTROLS IN MAIN WINDOW
// *****

// Supports Controls in Main window from old versions
HWND BLDCreateClientControlsDef(char *pTemplateName,DLGPROC lpNew)
{
    HWND      hDlg;

    hDlg = BLDCreateClientDlg(pTemplateName,MainhWnd,0,lpNew,BLDDLGCIENT,TRUE);
    if(hDlg)
    {
        hClient = hDlg;
        lpClient = lpNew;
        ShowWindow(hDlg,SW_SHOW);
    }
    return hDlg;
}

// Ensure that window is within screen.
void BLDMoveWindowDef(HWND hWnd,int x,int y,int nWidth,int nHeight,BOOL bRepaint)
{
    int      xMax,yMax,xNew,yNew;

    xMax = GetSystemMetrics(SM_CXSCREEN);
    yMax = GetSystemMetrics(SM_CYSCREEN);

    if ((nWidth<=xMax)&&(x+nWidth>xMax))
        xNew=xMax-nWidth;
    else
        xNew=x;

    if ((nHeight<=yMax)&&(y+nHeight>yMax))
        yNew=yMax-nHeight;
    else
        yNew=y;

    MoveWindow(hWnd,xNew,yNew,nWidth,nHeight,bRepaint);
    return;
}

void BLDMoveDlgClientDef(HWND ParenthWnd,HWND hNew)
{
    int      xLeft,yTop,xRight,yBottom;
    RECT      rClient,rMain,rDialog;
    int      dxDialog,dyDialog,dyExtra;
```



```
BLDCalcToolbarFrame(ParenthWnd, &xLeft, &yTop, &xRight, &yBottom);

GetClientRect(ParenthWnd, &rClient);
GetWindowRect(ParenthWnd, &rMain);
GetWindowRect(hNew, &rDialog);
if( (GetWindowLong(ParenthWnd, GWL_STYLE) & WS_CHILD) && GetParent(ParenthWnd))
{
    ScreenToClient(GetParent(ParenthWnd), (LPPOINT)&rMain.left);
    ScreenToClient(GetParent(ParenthWnd), (LPPOINT)&rMain.right);
}
dxDialog=(rDialog.right-rDialog.left)-(rClient.right-rClient.left);
dyDialog=(rDialog.bottom-rDialog.top)-(rClient.bottom-rClient.top);
if( (GetWindowLong(ParenthWnd, GWL_STYLE) & WS_CHILD) && GetParent(ParenthWnd))
    MoveWindow(ParenthWnd, rMain.left, rMain.top,
        (rMain.right-rMain.left)+dxDialog+xLeft+xRight,
        (rMain.bottom-rMain.top)+dyDialog+yTop+yBottom,
        TRUE);
else
    BLDMoveWindow(ParenthWnd, rMain.left, rMain.top,
        (rMain.right-rMain.left)+dxDialog+xLeft+xRight,
        (rMain.bottom-rMain.top)+dyDialog+yTop+yBottom,
        TRUE);
MoveWindow(hNew, xLeft, yTop,
    (rDialog.right-rDialog.left),
    (rDialog.bottom-rDialog.top),
    TRUE);
GetClientRect(ParenthWnd, &rClient);

// Compensate size if menu bar is more than one line.
if ((rDialog.bottom-rDialog.top+yTop+yBottom)>(rClient.bottom-rClient.top))
{
    dyExtra=(rDialog.bottom-rDialog.top)-(rClient.bottom-rClient.top);
    if( (GetWindowLong(ParenthWnd, GWL_STYLE) & WS_CHILD) && GetParent(ParenthWnd))
        MoveWindow(ParenthWnd, rMain.left, rMain.top,
            (rMain.right-rMain.left)+dxDialog+xLeft+xRight,
            (rMain.bottom-rMain.top)+dyDialog+yTop+yBottom+dyExtra,
            TRUE);
    else
        BLDMoveWindow(ParenthWnd, rMain.left, rMain.top,
            (rMain.right-rMain.left)+dxDialog+xLeft+xRight,
            (rMain.bottom-rMain.top)+dyDialog+yTop+yBottom+dyExtra,
            TRUE);
}
}
```

```
void BLDSetClientFocusDef(HWND hWnd)
{
    HWND      ChildhWnd;
    int       iRetval;

    ChildhWnd=GetWindow(hWnd, GW_CHILD);
    while(ChildhWnd)
    {
        iRetval=0;
        SendMessage(ChildhWnd, wBLDWindowType, 0, (LPARAM)(LPINT)&iRetval);
        if (iRetval == BLDDLGCIENT)
        {
            SetFocus(ChildhWnd);
            break;
        }
        ChildhWnd=GetWindow(ChildhWnd, GW_HWNDNEXT);
    }
}
```

```
    }
}

void BLDCClientMoveDef(HWND hWnd)
{
    HWND      ChildhWnd;
    int       iRetval;

    if ((GetActiveWindow() != hWnd) && (GetActiveWindow() != MainhWnd))
        return;

    ChildhWnd = GetWindow(hWnd, GW_CHILD);
    while(ChildhWnd)
    {
        iRetval = 0;
        SendMessage(ChildhWnd, wBLDWindowType, 0, (LPARAM) (LPINT) &iRetval);
        switch(iRetval)
        {
            case BLDTOOLBARTOP:
            case BLDTOOLBARBOTTOM:
            case BLDTOOLBARLEFT:
            case BLDTOOLBARRIGHT:
            case BLDDLGCIENT:
                // Hides the list box of a combo box
                SetFocus(ChildhWnd);
                break;
        }
        ChildhWnd = GetWindow(ChildhWnd, GW_HWNDNEXT);
    }
}

// *****
//      FUNCTION FOR HANDLING TOOLBARS CONTROLS IN MAIN WINDOW
// *****

// Startup procedure for Toolbars
HWND BLDCreateClientDlgDef(char *pTemplateName, HWND ParenthWnd, UINT message,
                           DLGPROC lpNew, int Position, BOOL bToMainWnd)
{
    int             iRetval;
    HANDLE          hRes, hMem;
    LPBLD_DLGTEMPLATE lpDlg;
    DWORD           styleold, style;
    HWND            hNew, ChildhWnd;

    if(bToMainWnd)
        ParenthWnd = MainhWnd;

    if (!IsWindow(ParenthWnd))
        return 0;

    if (hMDIClient && (Position == BLDDLGCIENT) && (ParenthWnd == MainhWnd))
        return FALSE;

    iRetval = 0;
    SendMessage(ParenthWnd, wBLDWindowType, 0, (LPARAM) (LPINT) &iRetval);
    while(ParenthWnd && iRetval)
    {
        ParenthWnd = GetParent(ParenthWnd);
        iRetval = 0;
        SendMessage(ParenthWnd, wBLDWindowType, 0, (LPARAM) (LPINT) &iRetval);
    }
}
```

```
    }

    if (!IsWindow(ParenthWnd))
        return FALSE;

    ChildhWnd=GetWindow(ParenthWnd,GW_CHILD);
    while(ChildhWnd)
    {
        iRetval=0;
        SendMessage(ChildhWnd,wBLDWindowType,0,(LPARAM)(LPINT)&iRetval);
        if (iRetval == Position)
        {
            DestroyWindow(ChildhWnd);
            break;
        }
        ChildhWnd=GetWindow(ChildhWnd,GW_HWNDNEXT);
    }

    // Get access to data structure of dialog box containing layout of controls
    hRes=FindResource(hInst,(LPSTR)pTemplateName,RT_DIALOG);
    if (!hRes)
        return 0;
    hMem=LoadResource(hInst,hRes);
    if (!hMem)
        return 0;
    lpDlg=(LPBLD_DLGTEMPLATE) LockResource(hMem);
    if (!lpDlg)
        return 0;

    styleold      = lpDlg->style;

    switch(Position)
    {
    case BLDTOOLBARTOP:
    case BLDTOOLBARBOTTOM:
        // Change dialog box data structure so it can be used as a window in client area
        style      = lpDlg->style&(TOOLBARSTRIP);
        lpDlg->style = lpDlg->style^style;
        lpDlg->style = lpDlg->style | WS_CHILD | WS_CLIPSIBLINGS;
#ifdef WIN32
        hNew = CreateDialogIndirect(hInst,(LPCDLGTEMPLATE)lpDlg,ParenthWnd,lpNew);
#else
        hNew = CreateDialogIndirect(hInst,(LPSTR)lpDlg,ParenthWnd,lpNew);
#endif
        if (!hNew)
            return 0;
        break;
    case BLDTOOLBARLEFT:
    case BLDTOOLBARRIGHT:
        // Change dialog box data structure so it can be used as a window in client area
        style      = lpDlg->style&(TOOLBARSTRIP);
        lpDlg->style = lpDlg->style^style;
        lpDlg->style = lpDlg->style | WS_CHILD | WS_CLIPSIBLINGS;
#ifdef WIN32
        hNew = CreateDialogIndirect(hInst,(LPCDLGTEMPLATE)lpDlg,ParenthWnd,lpNew);
#else
        hNew = CreateDialogIndirect(hInst,(LPSTR)lpDlg,ParenthWnd,lpNew);
#endif
        if (!hNew)
            return 0;
        break;
    case BLDDLGCCLIENT:
        style      = lpDlg->style&(CLIENTSTRIP);
        lpDlg->style = lpDlg->style^style;
```



```
        lpDlg->style = lpDlg->style | WS_CHILD | WS_CLIPSIBLINGS;
#ifdef WIN32
        hNew = CreateDialogIndirect(hInst, (LPCDLGTEMPLATE)lpDlg, ParenthWnd, lpNew);
#else
        hNew = CreateDialogIndirect(hInst, (LPSTR)lpDlg, ParenthWnd, lpNew);
#endif
        if (!hNew)
            return 0;
        // Move and size window in client area and main window
        BLDMoveDlgClient(ParenthWnd, hNew);
        SetFocus(hNew);
        break;
    }
    // Restore dialog box data structure.
    lpDlg->style = styleold;

    UnlockResource(hMem);
    FreeResource(hMem);

    if(message == WM_COMMAND && BLDDLCLIENT != Position)
    {
        RECT Rect;
        LPARAM lParam;

        ChildhWnd=GetWindow(ParenthWnd, GW_CHILD);
        while(ChildhWnd)
        {
            iRetval=0;
            SendMessage(ChildhWnd, wBLDWindowType, 0, (LPARAM) (LPINT) &iRetval);
            if (iRetval == BLDDLCLIENT)
            {
                BLDMoveDlgClient(ParenthWnd, ChildhWnd);
                break;
            }
            ChildhWnd=GetWindow(ChildhWnd, GW_HWNDNEXT);
        }
        GetClientRect(ParenthWnd, (LPRECT) &Rect);
        lParam=MAKELONG(Rect.right-Rect.left, Rect.bottom-Rect.top);
        PostMessage(ParenthWnd, WM_SIZE, SIZE_RESTORED, lParam);
    }
    return hNew;
}

void BLDCalcToolbarFrameDef(HWND hWnd, int *pxLeft, int *pyTop, int *pxRight, int *pyBottom)
{
    HWND        ChildhWnd;
    int         iRetval;
    RECT        Rect;

    *pxLeft=*pyTop=*pxRight=*pyBottom=0;

    ChildhWnd=GetWindow(hWnd, GW_CHILD);
    while(ChildhWnd)
    {
        iRetval=0;
        SendMessage(ChildhWnd, wBLDWindowType, 0, (LPARAM) (LPINT) &iRetval);
        GetWindowRect(ChildhWnd, (LPRECT) &Rect);
        ScreenToClient(hWnd, (LPPOINT) &Rect.left);
        ScreenToClient(hWnd, (LPPOINT) &Rect.right);
        switch(iRetval)
        {
            case BLDTOOLBARTOP:
                *pyTop=Rect.bottom-Rect.top;
        }
    }
}
```

```
        break;
    case BLDTOOLBARBOTTOM:
        *pyBottom=Rect.bottom-Rect.top;
        break;
    case BLDTOOLBARLEFT:
        *pxLeft=Rect.right-Rect.left;
        break;
    case BLDTOOLBARRIGHT:
        *pxRight=Rect.right-Rect.left;
        break;
    }
    ChildhWnd=GetWindow(ChildhWnd,GW_HWNDNEXT);
}
}
```

```
LRESULT BLDSIZEToolBarsDef(HWND hWnd,UINT message,int nSizeType,
                           int nWidth,int nHeight,BOOL VerticalOnTop)
{
    HWND      ChildhWnd;
    int        iRetval;
    RECT       Rect;
    int        xLeft,yTop,xRight,yBottom;

    if((nSizeType != SIZE_RESTORED)&&(nSizeType != SIZE_MAXIMIZED))
        return 0L;

    BLDCalcToolbarFrame(hWnd,&xLeft,&yTop,&xRight,&yBottom);

    ChildhWnd=GetWindow(hWnd,GW_CHILD);
    while(ChildhWnd)
    {
        iRetval=0;
        SendMessage(ChildhWnd,wBLDWindowType,0,(LPARAM)(LPINT)&iRetval);
        GetWindowRect(ChildhWnd,(LPRECT)&Rect);
        ScreenToClient(hWnd,(LPPOINT)&Rect.left);
        ScreenToClient(hWnd,(LPPOINT)&Rect.right);

        switch(iRetval)
        {
        case BLDTOOLBARTOP:
            Rect.bottom-=Rect.top;
            if(!VerticalOnTop)
            {
                Rect.left  =xLeft;
                Rect.right =nWidth-xLeft-xRight;
            }
            else
            {
                Rect.left  =0;
                Rect.right =nWidth;
            }
            Rect.top  =0;
            MoveWindow(ChildhWnd,Rect.left,Rect.top,Rect.right,
                      Rect.bottom,TRUE);
            InvalidateRect(ChildhWnd,(LPRECT)NULL,TRUE);
            break;
        case BLDTOOLBARBOTTOM:
            Rect.bottom-=Rect.top;
            if(!VerticalOnTop)
            {
                Rect.left  =xLeft;
                Rect.right =nWidth-xLeft-xRight;
            }
        }
```

```
        else
        {
            Rect.left  =0;
            Rect.right =nWidth;
        }
        Rect.top  =nHeight-Rect.bottom;
        MoveWindow (ChildhWnd,Rect.left,Rect.top,Rect.right,
                    Rect.bottom,TRUE);
        InvalidateRect (ChildhWnd, (LPRECT)NULL,TRUE);
        break;
    case BLDTOOLBARLEFT:
        Rect.right -=Rect.left;
        Rect.left  =0;
        if (VerticalOnTop)
        {
            Rect.top  =yTop;
            Rect.bottom =nHeight-yTop-yBottom;
        }
        else
        {
            Rect.top  =0;
            Rect.bottom =nHeight;
        }
        MoveWindow (ChildhWnd,Rect.left,Rect.top,Rect.right,
                    Rect.bottom,TRUE);
        InvalidateRect (ChildhWnd, (LPRECT)NULL,TRUE);
        break;
    case BLDTOOLBARRIGHT:
        Rect.right -=Rect.left;
        Rect.left  =nWidth-Rect.right;
        if (VerticalOnTop)
        {
            Rect.top  =yTop;
            Rect.bottom =nHeight-yTop-yBottom;
        }
        else
        {
            Rect.top  =0;
            Rect.bottom =nHeight;
        }
        MoveWindow (ChildhWnd,Rect.left,Rect.top,Rect.right,
                    Rect.bottom,TRUE);
        InvalidateRect (ChildhWnd, (LPRECT)NULL,TRUE);
        break;
    }
    ChildhWnd=GetWindow (ChildhWnd,GW_HWNDNEXT);
}
if (hMDIClient && hWnd == MainhWnd)
{
    GetClientRect (hWnd, (LPRECT)&Rect);
    Rect.left  +=xLeft;
    Rect.top   +=yTop;
    Rect.right -=xRight;
    Rect.bottom-=yBottom;
    MoveWindow (hMDIClient,Rect.left,Rect.top,Rect.right-Rect.left,
                Rect.bottom-Rect.top,TRUE);
}
return TRUE;
}
```

```
// *****
//          FUNCTION SENDING MDI MESSAGES
// *****
```



```
BOOL BLDSendMDIMessageDef(HWND hWnd,UINT message)
{
    if(hMDIClient)
    {
        if(message == WM_MDIGETACTIVE)
        {
            HWND hWnd;
#ifdef WIN32
            hWnd = (HWND) (SendMessage(hMDIClient,WM_MDIGETACTIVE,0,0L));
#else
            hWnd = (HWND) LOWORD (SendMessage(hMDIClient,WM_MDIGETACTIVE,0,0L));
#endif
            if(hWnd)
                SendMessage(hMDIClient,WM_MDIDESTROY, (WPARAM) hWnd, 0L);
            return TRUE;
        }
        SendMessage(hMDIClient,message,0,0L);
        return TRUE;
    }
    return FALSE;
}
```

```
// *****
//          FUNCTION EXPANDING OF DIALOG BOX
// *****
```

```
BOOL BLDSizedDlgDef(HWND hWnd,int xRightOf,int yBelow)
{
    RECT      Rect;
    RECT      OldWindowRect;
    RECT      NewActualClientRect;
    int       Cx,Cy;
    HWND      ClienthWnd;
    int       iRetVal;

    BLDFindCtrlsRightBottom(hWnd,&Cx,&Cy);

    Cx += xRightOf;
    Cy += yBelow;

    GetClientRect(hWnd,(LPRECT)&Rect);
    if (Rect.right >= Cx && Rect.bottom >= Cy )
        return TRUE;

    if(Rect.right < Cx)
        Rect.right = Cx;
    if(Rect.bottom < Cy)
        Rect.bottom = Cy;

    GetWindowRect(hWnd,&OldWindowRect);

    if(!(GetWindowLong(hWnd,GWL_STYLE) & WS_CHILD))
        ClientToScreen(hWnd,(LPPOINT)&Rect.left);
    else
        ScreenToClient(GetParent(hWnd),(LPPOINT)&OldWindowRect.left);

    Rect.right +=Rect.left;
    Rect.bottom+=Rect.top;

    AdjustWindowRectEx(&Rect,GetWindowLong(hWnd,GWL_STYLE),(BOOL)GetMenu(hWnd),
        GetWindowLong(hWnd,GWL_EXSTYLE));
}
```

```
if(OldWindowRect.top != Rect.top)
{
    Rect.bottom += OldWindowRect.top - Rect.top;
    Rect.top     =OldWindowRect.top;
}
if(OldWindowRect.left != Rect.left)
{
    Rect.right += OldWindowRect.left - Rect.left;
    Rect.left  =OldWindowRect.left;
}

MoveWindow (hWnd,Rect.left,Rect.top,Rect.right-Rect.left,
            Rect.bottom-Rect.top,TRUE);
GetClientRect (hWnd,&NewActualClientRect);

if(NewActualClientRect.bottom != Cy)
{
    Rect.bottom -= NewActualClientRect.bottom - Cy;
    MoveWindow (hWnd,Rect.left,Rect.top,Rect.right-Rect.left,
                Rect.bottom-Rect.top,TRUE);
}
if(NewActualClientRect.right != Cx)
{
    Rect.right -= NewActualClientRect.right - Cx;
    MoveWindow (hWnd,Rect.left,Rect.top,Rect.right-Rect.left,
                Rect.bottom-Rect.top,TRUE);
}

iRetval=0;
SendMessage (hWnd,wBLDWindowType,0,(LPARAM) (LPINT) &iRetval);

switch(iRetval)
{
case BLDTOOLBARTOP:
case BLDTOOLBARBOTTOM:
case BLDTOOLBARLEFT:
case BLDTOOLBARRIGHT:
    ClienthWnd=GetWindow (GetParent (hWnd),GW_CHILD);
    while (ClienthWnd)
    {
        iRetval=0;
        SendMessage (ClienthWnd,wBLDWindowType,0,(LPARAM) (LPINT) &iRetval);
        if (iRetval == BLDDLGCIENT)
            break;
        ClienthWnd=GetWindow (ClienthWnd,GW_HWNDNEXT);
    }
    if (ClienthWnd)
    {
        BLDMoveDlgClient (GetParent (hWnd),ClienthWnd);
    }
    break;
case BLDDLGCIENT:
    ClienthWnd=GetWindow (GetParent (hWnd),GW_CHILD);
    while (ClienthWnd)
    {
        iRetval=0;
        SendMessage (ClienthWnd,wBLDWindowType,0,(LPARAM) (LPINT) &iRetval);
        if (iRetval == BLDDLGCIENT)
            break;
        ClienthWnd=GetWindow (ClienthWnd,GW_HWNDNEXT);
    }
    if (ClienthWnd)
    {
        BLDMoveDlgClient (GetParent (hWnd),ClienthWnd);
    }
}
```

```
    }
    }
    return TRUE;
}

// *****
//      FUNCTIONS FOR CONTROLS AND FONT
// *****

BOOL BLDInitCtrlFontDef(HWND hDlg,int iCtrlId,int nHeight,int nWidth,int nEscapement,
                        int nOrientation,int fnWeight,BYTE fbItalic,BYTE fbUnderline,
                        BYTE fbStrikeOut,BYTE fbCharSet,BYTE fbOutputPrecision,
                        BYTE fbClipPrecision,BYTE fbQuality,BYTE fbPitchAndFamily,
                        char *lpszFace)
{
    HFONT      hFont;
    HWND       CtrlhWnd;

    CtrlhWnd=GetDlgItem(hDlg,iCtrlId);
    if(!CtrlhWnd)
        return FALSE;
    hFont=CreateFont(nHeight, nWidth, nEscapement,
                    nOrientation,fnWeight, fbItalic, fbUnderline,
                    fbStrikeOut, fbCharSet,fbOutputPrecision,
                    fbClipPrecision, fbQuality,
                    fbPitchAndFamily, lpszFace);

    if(hFont)
    {
        SetProp(CtrlhWnd,"BLDCTRLPROPHFONT",hFont);
        SendDlgItemMessage(hDlg,iCtrlId,WM_SETFONT,(WPARAM)hFont,(LPARAM)TRUE);
        return TRUE;
    }
    return FALSE;
}

BOOL BLDExitCtrlFontDef(HWND hDlg,int iCtrlId)
{
    HFONT      hFont;
    HWND       CtrlhWnd;

    CtrlhWnd=GetDlgItem(hDlg,iCtrlId);
    if (!CtrlhWnd)
        return FALSE;
    hFont=GetProp(CtrlhWnd,"BLDCTRLPROPHFONT");
    if(hFont)
    {
        SendDlgItemMessage(hDlg,iCtrlId,WM_SETFONT,(WPARAM)0,(LPARAM)FALSE);
        DeleteObject(hFont);
        RemoveProp(CtrlhWnd,"BLDCTRLPROPHFONT");
        return TRUE;
    }
    return FALSE;
}

// *****
//      FUNCTIONS DIALOG BOX AND CONTROLS BACKGROUND
// *****

HBRUSH BLDctlColorBrushSetOrgDef(HWND hWnd,HDC hDC)
{

```



```
    HBRUSH      hBrush;

    hBrush = GetProp(hWnd, "BLDPROPHBRUSH");
    if(hBrush)
    {
        int x,y,xScrolled,yScrolled;
#ifdef WIN32
        POINT p;
#else
        DWORD dwOldBrushOrg;
#endif

        UnrealizeObject(hBrush);

#ifdef WIN32
        GetBrushOrgEx(hDC, &p);
        x=p.x;
        y=p.y;
#else
        dwOldBrushOrg=GetBrushOrg(hDC);
        x=LOWORD(dwOldBrushOrg);
        y=HIWORD(dwOldBrushOrg);
#endif

        BLDGetDlgScrolled(hWnd, &xScrolled, &yScrolled);
        xScrolled=-xScrolled+x;
        yScrolled=-yScrolled+y;
#ifdef WIN32
        SetBrushOrgEx(hDC, xScrolled, yScrolled, NULL);
#else
        SetBrushOrg(hDC, xScrolled, yScrolled);
#endif
        SelectObject(hDC, hBrush);
    }
    return hBrush;
}
```

```
BOOL BLDInitSolidBrushDef(HWND hWnd, COLORREF ColorRef)
{
    HBRUSH      hBrush;

    if (!hWnd)
        return FALSE;
    hBrush=CreateSolidBrush(ColorRef);
    if(hBrush)
    {
        SetProp(hWnd, "BLDPROPHBRUSH", hBrush);
        return TRUE;
    }
    return FALSE;
}
```

```
BOOL BLDInitPatternBrushDef(HWND hWnd, char *pBitmapName)
{
    HBRUSH      hBrush;
    HBITMAP     hBitmap;

    if (!hWnd)
        return FALSE;
    hBitmap = BLDLoadBitmap(hInst, pBitmapName);
    if(hBitmap)
```

```
{
    hBrush = CreatePatternBrush(hBitmap);
    DeleteObject(hBitmap);
    if(hBrush)
    {
        SetProp(hWnd, "BLDPROPHBRUSH", hBrush);
        return TRUE;
    }
}
return FALSE;
}
```

BOOL BLDExitBrushDef(HWND hWnd)

```
{
    HBRUSH      hBrush;

    if (!hWnd)
        return FALSE;
    hBrush=GetProp(hWnd, "BLDPROPHBRUSH");
    if (hBrush)
    {
        DeleteObject(hBrush);
        RemoveProp(hWnd, "BLDPROPHBRUSH");
        return TRUE;
    }
    return FALSE;
}
```

HBRUSH BLDctlColorStockBrushDef(HWND hWnd,int fnObject)

```
{
    HBRUSH      hBrush;

    hBrush=GetStockObject(fnObject);
    if(hBrush)
        return hBrush;
    return BLDctlColorDefaultBrush(hWnd);
}
```

HBRUSH BLDctlColorPropBrushDef(HWND hWnd)

```
{
    HBRUSH      hBrush;

    hBrush=GetProp(hWnd, "BLDPROPHBRUSH");
    if(hBrush)
        return hBrush;
    return BLDctlColorDefaultBrush(hWnd);
}
```

HBRUSH BLDctlColorDefaultBrushDef(HWND hWnd)

```
{
    HBRUSH      hBrush;

    hBrush      = BLDGetGlobalBrushDef(hWnd, 0);
    if (hBrush)
        return hBrush;
}
```

```
#ifdef WIN32
    hBrush=(HBRUSH)GetClassLong(hWnd,GCL_HBRBACKGROUND);
#else
    hBrush=(HBRUSH)GetClassWord(hWnd,GCW_HBRBACKGROUND);
#endif
    if(hBrush)
        return hBrush;
    hBrush = GetStockObject(WHITE_BRUSH);
    if(hBrush)
        return hBrush;
    return (HBRUSH)0;
}

// *****
//      FUNCTIONS FOR HELP HANDLING
// *****

static BOOL bHelp=FALSE;

BOOL BLDCheckF1HelpKeyDef(BOOL bShift)
{
    if(GetKeyState(VK_F1) >= 0)
        return FALSE;
    if(bShift)
    {
        if(GetKeyState(VK_SHIFT) >= 0)
            return FALSE;
    }
    else
    {
        if(GetKeyState(VK_SHIFT) < 0)
            return FALSE;
    }
    if(GetKeyState(VK_CONTROL) < 0)
        return FALSE;
    if(GetKeyState(VK_MENU) < 0)
        return FALSE;
    return TRUE;
}

void BLDHelpTranslationDef(MSG *pmsg)
{
    if(pmsg->message == WM_KEYDOWN)
    {
        if(BLDCheckF1HelpKey(TRUE))
        {
            if(GetMenu(GetActiveWindow()))
            {
                HCURSOR hCursor;
                bHelp=TRUE;
                hCursor=LoadCursor(hInst,"HELP");
                if(hCursor)
                {
                    SetCursor(hCursor);
                }
            }
            return;
        }
        if(BLDCheckF1HelpKey(FALSE))
        {

```



```
    HWND hActiveWnd;
    WPARAM bFromMDIWnd;

    bHelp=FALSE;
    hActiveWnd=GetActiveWindow();
    bFromMDIWnd=FALSE;
    if(hMDIClient)
    {
        if(hActiveWnd == MainhWnd)
        {
            HWND hActiveMDIWnd;

#ifdef WIN32
            hActiveMDIWnd=(HWND) (SendMessage(hMDIClient,
                WM_MDIGETACTIVE,0,0L));
#else
            hActiveMDIWnd=(HWND) LOWORD(SendMessage(hMDIClient,
                WM_MDIGETACTIVE,0,0L));
#endif

            if(hActiveMDIWnd)
            {
                hActiveWnd=hActiveMDIWnd;
                bFromMDIWnd=TRUE;
            }
        }
    }
    PostMessage(hActiveWnd,wHelpMessage,bFromMDIWnd,0L);
    return;
}

}

void BLDShowHelpDef(HWND hWnd,UINT fuCommand,DWORD dwData)
{
    char        szWinHelpFile[BLD_MAXPATH];

    BLDGetHelpFileName(szWinHelpFile);
    SetCursor(LoadCursor(NULL,IDC_WAIT));
    WinHelp(hWnd,(LPSTR) szWinHelpFile,fuCommand,dwData);
}

BOOL BLDHelpFilterDef(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam,
    DWORD dwHelpId,LPARAM *plRetval,BOOL bFromDlg)
{
    HCURSOR      hCursor;
    HWND         hCtrl;
    WORD         notification;
    HDC          hDC;
    static HBRUSH hGray = 0;

    if (dwDialogProp&BLDGRAY_DIALOGBOX)
    {
#ifdef WIN32
        switch (message)
        {
            case WM_CTLCOLORLISTBOX:
            case WM_CTLCOLORMSGBOX:
                hCtrl      = (HWND) lParam;
                hDC        = (HDC) wParam;
                *plRetval   = (LPARAM) BLDGetGlobalBrushDef(hCtrl,hDC);
                return FALSE;
                break;
            case WM_CTLCOLOREDIT:

```

```
        if (!(dwDialogProp&BLDGRAY_EDIT))
            return FALSE;
        break;
    case WM_CTLCOLORBTN:
        if (!(dwDialogProp&BLDGRAY_BUTTON))
            return FALSE;
        break;
    case WM_CTLCOLORSCROLLBAR:
        if (!(dwDialogProp&BLDGRAY_SCROLLBAR))
            return FALSE;
        break;
    case WM_CTLCOLORSTATIC:
        if (!(dwDialogProp&BLDGRAY_TEXT))
            return FALSE;
        break;
    case WM_CTLCOLORDLG:
        break;
    default:
        goto HELPHANDLING;
        break;
    }
#else
    switch (message)
    {
    case WM_CTLCOLOR:
        switch ((int) HIWORD(lParam))
        {
        case CTLCOLOR_LISTBOX:
        case CTLCOLOR_MSGBOX:
            hCtrl      = (HWND) LOWORD(lParam);
            hDC         = (HDC) wParam;
            *plRetval    = BLDGetGlobalBrushDef(hCtrl,hDC);
            return FALSE;
            break;
        case CTLCOLOR_EDIT:
            if (!(dwDialogProp&BLDGRAY_EDIT))
                return FALSE;
            break;
        case CTLCOLOR_BTN:
            if (!(dwDialogProp&BLDGRAY_BUTTON))
                return FALSE;
            break;
        case CTLCOLOR_SCROLLBAR:
            if (!(dwDialogProp&BLDGRAY_SCROLLBAR))
                return FALSE;
            break;
        case CTLCOLOR_STATIC:
            if (!(dwDialogProp&BLDGRAY_TEXT))
                return FALSE;
            break;
        case CTLCOLOR_DLG:
            break;
        default:
            return FALSE;
            break;
        }
        break;
    default:
        goto HELPHANDLING;
        break;
    }

#endif
    hDC      = (HDC) wParam;
```

```
SetBkColor(hDC,RGB(192,192,192));
if (!hGray)
    hGray = GetStockObject(LTGRAY_BRUSH);
*plRetval = (LPARAM)hGray;
return FALSE; // Don't stop other processing
}
```

## HELPHANDLING:

```
if (!bHelpSupport)
    return FALSE;

if (message==wHelpMessage)
{
    if(dwHelpId)
    {
        BLDSHOWHELP(hWnd,HELP_CONTEXT,dwHelpId);
        return TRUE;
    }
    else
    {
        if(wParam) // It is a MDI Child window with no dwHelpId
            PostMessage(MainhWnd,wHelpMessage,0,0L);
    }
}

switch(message)
{
case WM_DESTROY:
    if(hWnd == MainhWnd)
        BLDSHOWHELP(hWnd,HELP_QUIT,0);
    break;
case WM_COMMAND:
    if(bHelp)
    {
        // Extracting data from message
        hCtrl = (HWND)(UINT)lParam;
#ifdef WIN32
        notification = HIWORD(wParam);
#else
        notification = HIWORD(lParam);
#endif
        bHelp=FALSE;
        if(!(GetKeyState(VK_ESCAPE) & 0x8000 ))
        {
            if(!hCtrl&&(notification==0||notification==1)) //Menu or Accelerator
                BLDMenuHelp(hWnd,message,wParam,lParam);
            else
                PostMessage(hWnd,wHelpMessage,0,0L);
        }
#ifdef WIN32
        SetCursor((HCURSOR)GetClassLong(GetActiveWindow(),GCL_HCURSOR));
#else
        SetCursor((HCURSOR)GetClassWord(GetActiveWindow(),GCW_HCURSOR));
#endif
        *plRetval=TRUE;
        return TRUE;
    }
    break;
case WM_ACTIVATE:
    bHelp=FALSE;
    break;
case WM_SETCURSOR:
    if(bHelp)
```



```
{
hCursor=LoadCursor(hInst,"HELP");
if(hCursor)
{
SetCursor(hCursor);
*plRetval=TRUE;
if(bFromDlg)
SetWindowLong(hWnd,DWL_MSGRESULT,MAKELONG(TRUE, 0));
return TRUE;
}
}
break;
case WM_KEYDOWN:
if(wParam == VK_ESCAPE && bHelp)
{
bHelp = FALSE;
#ifdef WIN32
SetCursor((HCURSOR)GetClassLong(GetActiveWindow(),GCL_HCURSOR));
#else
SetCursor((HCURSOR)GetClassWord(GetActiveWindow(),GCW_HCURSOR));
#endif
}
break;
case WM_ENTERIDLE:
if ((wParam == MSGF_MENU) && BLDCheckFlHelpKey(FALSE))
{
bHelp = TRUE;
PostMessage(GetActiveWindow(), WM_KEYDOWN, VK_RETURN, 0L);
*plRetval=FALSE;
return TRUE;
}
if(BLDCheckFlHelpKey(TRUE))
{
if(GetMenu(GetActiveWindow()))
{
bHelp=TRUE;
hCursor=LoadCursor(hInst,"HELP");
if(hCursor)
{
SetCursor(hCursor);
}
}
*plRetval=FALSE;
return TRUE;
}
if(BLDCheckFlHelpKey(FALSE))
{
bHelp=FALSE;
PostMessage(GetActiveWindow(),wHelpMessage,0,0L);
*plRetval=FALSE;
return TRUE;
}
if(bHelp && (wParam == MSGF_DIALOGBOX) )
{
if(GetKeyState(VK_ESCAPE) & 0x8000 )
{
bHelp = FALSE;
#ifdef WIN32
SetCursor((HCURSOR)GetClassLong(GetActiveWindow(),GCL_HCURSOR));
#else
SetCursor((HCURSOR)GetClassWord(GetActiveWindow(),GCW_HCURSOR));
#endif
}
*plRetval=FALSE;
}
```

```
        return TRUE;
    }
    break;
}

return FALSE;
}
```

HBUSH BLDGetGlobalBrushDef(HWND hCtrl,HDC hDC)

```
{
    char    szClass[21];
    static HBUSH hGray = 0;

    if (!hCtrl)
        return 0;

    if (!hGray)
        hGray = GetStockObject(LTGRAY_BRUSH);

    if (!GetClassName(hCtrl,szClass,20))
        return 0;

    if (lstrcmpi(szClass,"EDIT")==0)
    {
        if (dwDialogProp&BLDGRAY_EDIT)
            goto RETGRAYBRUSH;
        else
            return 0;
    }
    if (lstrcmpi(szClass,"COMBOBOX")==0)
    {
        if (dwDialogProp&BLDGRAY_COMBOBOX)
            goto RETGRAYBRUSH;
        else
            return 0;
    }
    if (lstrcmpi(szClass,"LISTBOX")==0)
    {
        if (dwDialogProp&BLDGRAY_LISTBOX)
            goto RETGRAYBRUSH;
        else
            return 0;
    }
    if (lstrcmpi(szClass,"BUTTON")==0)
    {
        if ((dwDialogProp&BLDGRAY_BUTTON))
            goto RETGRAYBRUSH;
        else
            return 0;
    }
    if (lstrcmpi(szClass,"SCROLLBAR")==0)
    {
        if (dwDialogProp&BLDGRAY_SCROLLBAR)
            goto RETGRAYBRUSH;
        else
            return 0;
    }
    if (lstrcmpi(szClass,"STATIC")==0)
    {
        if (dwDialogProp&BLDGRAY_TEXT)
            goto RETGRAYBRUSH;
        else
            return 0;
    }
}
```

```
    }

    return 0;

RETGRAYBRUSH:
    if (hDC)
        SetBkColor(hDC, RGB(192,192,192));
    return hGray;
}

LRESULT BLDDefWindowProcMsgDef(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    if(!hMDIClient)
        return DefWindowProc(hWnd, message, wParam, lParam);
    if(hWnd == MainhWnd)
        return DefFrameProc(hWnd, hMDIClient, message, wParam, lParam);
    if(GetParent(hWnd) == hMDIClient)
        return DefMDIChildProc(hWnd, message, wParam, lParam);
    return DefWindowProc(hWnd, message, wParam, lParam);
}

// *****
//          FUNCTIONS AND GLOBAL VARIABLES FOR HANDLING
//          OF MULTIPLE INSTANCES OF TOOLBARS AND
//          CLIENT AREA CONTROLS
// *****

static HANDLE  BLDhClientDlgMem  = 0;
static int     BLDhClientDlgCount = 0;

typedef struct tagBLDCLIENTITEM
{
    HWND        hDlg;
    DLGPROC     lpProc;
}BLDCLIENTITEM;

BOOL BLDAddClientDlgDef(HWND hDlg,DLGPROC lpProc)
{
    BLDCLIENTITEM FAR *lpClientItem;

    if(!BLDhClientDlgMem)
        BLDhClientDlgMem=GlobalAlloc(GMEM_MOVEABLE,1);

    BLDhClientDlgMem=GlobalReAlloc(BLDhClientDlgMem,(BLDhClientDlgCount +2)*sizeof(BLDCLIENTITEM),GMEM_MOVEABLE);

    if(!BLDhClientDlgMem)
        return FALSE;
    lpClientItem =(BLDCLIENTITEM FAR *)GlobalLock(BLDhClientDlgMem);
    if(!lpClientItem)
        return FALSE;
    lpClientItem += BLDhClientDlgCount;
    lpClientItem->hDlg=hDlg;
    lpClientItem->lpProc=lpProc;
    GlobalUnlock(BLDhClientDlgMem);
    BLDhClientDlgCount++;
    return TRUE;
}
```



```
BOOL BLDRemoveClientDlgDef(HWND hDlg)
{
    BLDCLIENTITEM FAR *lpClientItem;
    BLDCLIENTITEM FAR *lpClientItem2;
    LPSTR      lpstr;
    int        iFound;
    int        i;

    lpstr = (LPSTR)GlobalLock(BLDhClientDlghMem);
    if(!lpstr)
        return FALSE;
    iFound = -1;
    for(i = 0; i < BLDhClientDlgCount && iFound == -1; i++)
    {
        lpClientItem = (BLDCIENTITEM FAR *)lpstr+i;
        if(lpClientItem->hDlg == hDlg)
        {
            FreeProcInstance((FARPROC)lpClientItem->lpProc);
            iFound = i;
        }
    }
    if(iFound == -1)
    {
        GlobalUnlock(BLDhClientDlghMem);
        return FALSE;
    }
    for(i = iFound+1; i < BLDhClientDlgCount; i++)
    {
        lpClientItem = (BLDCIENTITEM FAR *)lpstr+i;
        lpClientItem2 = (BLDCIENTITEM FAR *)lpstr+i-1;
        lpClientItem2->hDlg = lpClientItem->hDlg;
        lpClientItem2->lpProc = lpClientItem->lpProc;
    }
    GlobalUnlock(BLDhClientDlghMem);
    BLDhClientDlgCount--;
    if(BLDhClientDlgCount)
        BLDhClientDlghMem = GlobalReAlloc(BLDhClientDlghMem, (BLDhClientDlgCount)*sizeof(BLDCIENTITEM), GMEM_MOVEABLE);
    else
    {
        GlobalFree(BLDhClientDlghMem);
        BLDhClientDlghMem = 0;
    }
    return TRUE;
}

BOOL BLDIsClientDlgDialogMessageDef(MSG *pMsg)
{
    BLDCLIENTITEM FAR *lpClientItem;
    LPSTR      lpstr;
    int        i;

    if(!BLDhClientDlghMem)
        return FALSE;
    lpstr = (LPSTR)GlobalLock(BLDhClientDlghMem);
    if(!lpstr)
        return FALSE;

    for(i = 0; i < BLDhClientDlgCount ; i++)
    {
        lpClientItem = (BLDCIENTITEM FAR *)lpstr+i;
```

```
        if(IsDialogMessage(lpClientItem->hDlg,pMsg))
        {
            GlobalUnlock(BLDhClientDlghMem);
            return TRUE;
        }
    GlobalUnlock(BLDhClientDlghMem);
    return FALSE;
}
```

```
// Filename: USERCODE.WMC
// "EAMAT42" Generated by WindowsMAKER Professional.
// Author: Laura L. Downey

//
// *****
// Do not add code here. Add code in the .C file.
//
// This file is maintained by WindowsMAKER Professional.
// As you make changes in your application using WindowsMAKER Professional,
// this file is automatically updated, therefore you never modify this file.
//
//
// For more information,
// see the section "How code is generated" in the documentation.
//
// *****

// *****
// Modal Dialog Box: QUERY
// *****

// Startup procedure for modal dialog box
int BLD_QUERYDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC lpProc;
    int ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_QUERYDlgProc, hWnd);
    ReturnValue = DialogBox(hWnd, (LPSTR)(szDlgName?szDlgName:"QUERY"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue==--1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"QUERY"),
                          MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_QUERYDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL bRet;

    bRet = FALSE; // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch (message)
    {
    case WM_INITDIALOG:
        bRet = TRUE; // Default return for WM_INITDIALOG is TRUE
        BLDInitSolidBrush(hDlg, RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 109), RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 115), RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 119), RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 116), RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 121), RGB(192,192,192));
    }
```



```
BLDInitSolidBrush(GetDlgItem(hDlg,117),RGB(192,192,192));
break;
```

```
case WM_COMMAND:
```

```
{
    WORD      wId;
    WORD      notification;
    HWND      hCtrl;
```

```
    // Extracting data from message
    wId      = LOWORD(wParam);
    hCtrl    = (HWND) (UINT) lParam;
```

```
#ifdef WIN32
```

```
    notification = HIWORD(wParam);
```

```
#else
```

```
    notification = HIWORD(lParam);
```

```
#endif
```

```
    if(!hCtrl)          // Menu input or CR
    {
        if (BLDMenuCommand(hDlg,message,wParam,lParam))
            return TRUE;
    }
```

```
    switch(wId)
```

```
    {
```

```
    case 1:
```

```
        switch(notification)
```

```
        {
```

```
        case BN_CLICKED:
```

```
            BLD_OKDlgFunc(hDlg,message,wParam,lParam);
```

```
            EndDialog(hDlg,1);
```

```
            return TRUE;
```

```
            break;
```

```
        default:
```

```
            break;
```

```
        }
```

```
        break;
```

```
    case 2:
```

```
        switch(notification)
```

```
        {
```

```
        case BN_CLICKED:
```

```
            EndDialog(hDlg,2);
```

```
            return TRUE;
```

```
            break;
```

```
        default:
```

```
            break;
```

```
        }
```

```
        break;
```

```
    default:
```

```
        break;
```

```
    }
```

```
}
```

```
break;
```

```
case WM_DRAWITEM:
```

```
{
```

```
    LPDRAWITEMSTRUCT lpDrawItem;
```

```
    lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
```

```
    if (BLDDrawItem(hDlg,lpDrawItem))
```

```
        return TRUE;
```

```
}
```

```
break;
```

```
break;
```

```
case WM_DESTROY:
    BLDExitBrush(hDlg);
    BLDExitBrush(GetDlgItem(hDlg,109));
    BLDExitBrush(GetDlgItem(hDlg,115));
    BLDExitBrush(GetDlgItem(hDlg,119));
    BLDExitBrush(GetDlgItem(hDlg,116));
    BLDExitBrush(GetDlgItem(hDlg,121));
    BLDExitBrush(GetDlgItem(hDlg,117));
    break;

#ifdef WIN32
case WM_CTLCOLORMSGBOX:
case WM_CTLCOLOREDIT:
case WM_CTLCOLORLISTBOX:
case WM_CTLCOLORBTN:
case WM_CTLCOLORDLG:
case WM_CTLCOLORSCROLLBAR:
case WM_CTLCOLORSTATIC:
#else
case WM_CTLCOLOR:
#endif
    // Extracting data from message
    {
        HWND          hCtrl;

#ifdef WIN32
        hCtrl          = (HWND)lParam;
#else
        hCtrl          = (HWND)LOWORD(lParam);
#endif
#ifdef WIN32
        if(message == WM_CTLCOLORDLG)
            return (BOOL)BLDctlColorBrushSetOrg(hDlg, (HDC)wParam);
#else
        if(HIWORD(lParam) == CTLCOLOR_DLG)
            return (BOOL)BLDctlColorBrushSetOrg(hDlg, (HDC)wParam);
#endif
#ifdef WIN32
        switch(GetDlgItemID(hCtrl))
        {
            case 109:
                SetTextColor((HDC)wParam,RGB(0,0,128));
                SetBkMode((HDC)wParam,TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case 115:
                SetTextColor((HDC)wParam,RGB(0,0,128));
                SetBkMode((HDC)wParam,TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case 119:
                SetTextColor((HDC)wParam,RGB(0,0,128));
                SetBkMode((HDC)wParam,TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case 116:
                SetTextColor((HDC)wParam,RGB(0,0,128));
                SetBkMode((HDC)wParam,TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case 121:
                SetTextColor((HDC)wParam,RGB(0,0,128));
                SetBkMode((HDC)wParam,TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
        }
    }
}
```

```
        case 117:
            SetTextColor((HDC)wParam,RGB(0,0,128));
            SetBkMode((HDC)wParam,TRANSPARENT);
            return (BOOL)BLDCtlColorPropBrush(hCtrl);
            break;
        }
    }
    break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
// Controls in Main Window: MAIN
// *****

// Startup procedure for client dialog box
HWND BLD_MAINClFuncDef(HWND hWnd,char *szDlgName,UINT message)
{
    if(MAINhDlg&&IsWindow(MAINhDlg))
    {
        return MAINhDlg;
    }
    MAINlpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_MAINClProc,hInst);
    MAINhDlg = BLDCreateClientDlg((szDlgName?szDlgName:"MAIN"),
                                hWnd,message,MAINlpProc,BLDDLGCIENT,TRUE);
    if (MAINhDlg==0)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"MAIN"),
                           MB_OK | MB_ICONHAND);
    else
        ShowWindow(MAINhDlg,SW_SHOW);
    return MAINhDlg;
}

// Default dialog box procedure
BOOL BLD_MAINDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL      bRet;

    bRet      = FALSE;    // Default return value if not processed

    if(BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGCIENT,0,&bRet))
        return bRet;

    switch(message)
    {
        case WM_INITDIALOG:
            bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE

            if (GetParent(hDlg))
                BLDMoveDlgClient(GetParent(hDlg),hDlg);
            break;

        case WM_COMMAND:
            {
                WORD      wId;
                WORD      notification;
            }
    }
}
```



```
// Extracting data from message
wId      = LOWORD(wParam);
#ifdef WIN32
    notification = HIWORD(wParam);
#else
    notification = HIWORD(lParam);
#endif

switch(wId)
{
case 125:
    switch(notification)
    {
    case BN_CLICKED:
        BLD_QUERYDlgFunc(hDlg,message,wParam,lParam);
        return TRUE;
        break;
    default:
        break;
    }
    break;
case 126:
    switch(notification)
    {
    case BN_CLICKED:
        BLD_BrowseEntryDlgFunc(hDlg,message,wParam,lParam);
        return TRUE;
        break;
    default:
        break;
    }
    break;
case 127:
    switch(notification)
    {
    case BN_CLICKED:
        BLD_PrintDlgFunc(hDlg,message,wParam,lParam);
        return TRUE;
        break;
    default:
        break;
    }
    break;
case 128:
    switch(notification)
    {
    case BN_CLICKED:
        BLD_Function2DlgFunc(hDlg,message,wParam,lParam);
        return TRUE;
        break;
    default:
        break;
    }
    break;
case 130:
    switch(notification)
    {
    case BN_CLICKED:
        BLD_ReportDlgFunc(hDlg,message,wParam,lParam);
        return TRUE;
        break;
    default:
        break;
    }
}
```

```
        break;
    default:
        break;
    }
}
break;

case WM_DRAWITEM:
{
    LPDRAWITEMSTRUCT lpDrawItem;

    lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
    if (BLDDrawItem(hDlg,lpDrawItem))
        return TRUE;
    }
    break;
break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//                               Modal Dialog Box: PRINT
// *****

// Startup procedure for modal dialog box
int BLD_PrintDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_PrintDlgProc,hInst);
    ReturnValue = DialogBox(hInst, (LPSTR) (szDlgName?szDlgName:"PRINT"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"PRINT"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_PrintDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL          bRet;

    bRet          = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        BLDInitSolidBrush(hDlg,RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg,116),RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg,109),RGB(192,192,192));
    }
```

```
BLDInitSolidBrush(GetDlgItem(hDlg,115),RGB(192,192,192));
BLDInitSolidBrush(GetDlgItem(hDlg,118),RGB(192,192,192));
BLDInitSolidBrush(GetDlgItem(hDlg,121),RGB(192,192,192));
break;
```

```
case WM_COMMAND:
```

```
{
    WORD        wId;
    WORD        notification;
    HWND        hCtrl;
```

```
    // Extracting data from message
    wId          = LOWORD(wParam);
    hCtrl        = (HWND) (UINT) lParam;
```

```
#ifdef WIN32
```

```
    notification = HIWORD(wParam);
```

```
#else
```

```
    notification = HIWORD(lParam);
```

```
#endif
```

```
    if(!hCtrl)                // Menu input or CR
    {
        if (BLDMenuCommand(hDlg,message,wParam,lParam))
            return TRUE;
    }
```

```
    switch(wId)
```

```
    {
```

```
    case 1:
```

```
        switch(notification)
```

```
        {
```

```
            case BN_CLICKED:
```

```
                EndDialog(hDlg,1);
```

```
                return TRUE;
```

```
                break;
```

```
            default:
```

```
                break;
```

```
        }
```

```
        break;
```

```
    default:
```

```
        break;
```

```
    }
```

```
    }
```

```
    break;
```

```
case WM_DRAWITEM:
```

```
{
```

```
    LPDRAWITEMSTRUCT lpDrawItem;
```

```
    lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
```

```
    if (BLDDrawItem(hDlg,lpDrawItem))
```

```
        return TRUE;
```

```
    }
```

```
    break;
```

```
break;
```

```
case WM_DESTROY:
```

```
    BLDExitBrush(hDlg);
```

```
    BLDExitBrush(GetDlgItem(hDlg,116));
```

```
    BLDExitBrush(GetDlgItem(hDlg,109));
```

```
    BLDExitBrush(GetDlgItem(hDlg,115));
```

```
    BLDExitBrush(GetDlgItem(hDlg,118));
```

```
    BLDExitBrush(GetDlgItem(hDlg,121));
```

```
break;
```

```
#ifdef WIN32
```



```
case WM_CTLCOLORMSGBOX:
case WM_CTLCOLOREDIT:
case WM_CTLCOLORLISTBOX:
case WM_CTLCOLORBTN:
case WM_CTLCOLORDLG:
case WM_CTLCOLORSCROLLBAR:
case WM_CTLCOLORSTATIC:
#else
case WM_CTLCOLOR:
#endif
    // Extracting data from message
    {
        HWND          hCtrl;

#ifdef WIN32
        hCtrl          = (HWND)lParam;
#else
        hCtrl          = (HWND)LOWORD(lParam);
#endif
#ifdef WIN32
        if(message == WM_CTLCOLORDLG)
            return (BOOL)BLDctlColorBrushSetOrg(hDlg, (HDC)wParam);
#else
        if(HIWORD(lParam) == CTLCOLOR_DLG)
            return (BOOL)BLDctlColorBrushSetOrg(hDlg, (HDC)wParam);
#endif
#ifdef WIN32
        switch(GetDlgCtrlID(hCtrl))
        {
            case 116:
                SetTextColor((HDC)wParam, RGB(0,0,128));
                SetBkMode((HDC)wParam, TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case 109:
                SetTextColor((HDC)wParam, RGB(0,0,128));
                SetBkMode((HDC)wParam, TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case 115:
                SetTextColor((HDC)wParam, RGB(0,0,128));
                SetBkMode((HDC)wParam, TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case 118:
                SetTextColor((HDC)wParam, RGB(0,0,128));
                SetBkMode((HDC)wParam, TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case 121:
                SetTextColor((HDC)wParam, RGB(0,0,128));
                SetBkMode((HDC)wParam, TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
        }
        break;
#endif
default:
    break;
    }
return bRet;          // No explicit return - return default
}
```

```
// *****
//           Modal Dialog Box: QINFO
// *****

// Startup procedure for modal dialog box
int BLD_OKDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_OKDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"QINFO"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"QINFO"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_OKDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL          bRet;

    bRet          = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {
        case WM_INITDIALOG:
            bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
            BLDInitSolidBrush(hDlg, RGB(224, 223, 225));
            BLDInitSolidBrush(GetDlgItem(hDlg, 147), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 149), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 137), RGB(192, 192, 192));
            BLDInitCtrlFont(hDlg, 137, -12, 0, 0, 0, 700, 0, 0, 0, 0, 3, 2, 1, 34, "Arial");
            BLDInitSolidBrush(GetDlgItem(hDlg, 150), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 151), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 152), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 153), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 154), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 155), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 158), RGB(192, 192, 192));
            BLDInitCtrlFont(hDlg, 158, -11, 0, 0, 0, 700, 0, 0, 0, 0, 3, 2, 1, 34, "Arial");
            BLDInitSolidBrush(GetDlgItem(hDlg, 160), RGB(192, 192, 192));
            BLDInitCtrlFont(hDlg, 160, -11, 0, 0, 0, 700, 0, 0, 0, 0, 3, 2, 1, 34, "Arial");
            BLDInitSolidBrush(GetDlgItem(hDlg, 162), RGB(192, 192, 192));
            BLDInitCtrlFont(hDlg, 162, -11, 0, 0, 0, 700, 0, 0, 0, 0, 3, 2, 1, 34, "Arial");
            break;

        case WM_COMMAND:
            {
                WORD      wId;
                WORD      notification;
                HWND      hCtrl;

                // Extracting data from message
                wId        = LOWORD(wParam);
                hCtrl      = (HWND)(UINT)lParam;
            }
    }
}
```

```
#ifdef WIN32
    notification = HIWORD(wParam);
#else
    notification = HIWORD(lParam);
#endif
    if(!hCtrl)           // Menu input or CR
    {
        if (BLDMenuCommand(hDlg,message,wParam,lParam))
            return TRUE;
    }
    switch(wId)
    {
    case 124:
        switch(notification)
        {
            case BN_CLICKED:
                BLD_HeaderDetailDlgFunc(hDlg,message,wParam,lParam);
                return TRUE;
                break;
            default:
                break;
        }
        break;
    case ID_QRepTot:
        switch(notification)
        {
            case BN_CLICKED:
                BLD_ReportTotalsDlgFunc(hDlg,message,wParam,lParam);
                return TRUE;
                break;
            default:
                break;
        }
        break;
    case 159:
        switch(notification)
        {
            case BN_CLICKED:
                BLD_qparamDlgFunc(hDlg,message,wParam,lParam);
                return TRUE;
                break;
            default:
                break;
        }
        break;
    default:
        break;
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        switch(lpDrawItem->CtlID)
        {
            case ID_Change:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"ARR",TRUE);
                return TRUE;
                break;
        }
    }
}
```



```
        case 159:
            if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                if (lpDrawItem->CtlType==ODT_BUTTON)
                    BLDDDrawBitmap(lpDrawItem,"PARAM",TRUE);
            return TRUE;
            break;
        default:
            if (BLDDDrawItem(hDlg,lpDrawItem))
                return TRUE;
            break;
    }
}
break;

case WM_DESTROY:
    BLDExitBrush(hDlg);
    BLDExitBrush(GetDlgItem(hDlg,147));
    BLDExitBrush(GetDlgItem(hDlg,149));
    BLDExitCtrlFont(hDlg,137);
    BLDExitBrush(GetDlgItem(hDlg,137));
    BLDExitBrush(GetDlgItem(hDlg,150));
    BLDExitBrush(GetDlgItem(hDlg,151));
    BLDExitBrush(GetDlgItem(hDlg,152));
    BLDExitBrush(GetDlgItem(hDlg,153));
    BLDExitBrush(GetDlgItem(hDlg,154));
    BLDExitBrush(GetDlgItem(hDlg,155));
    BLDExitBrush(GetDlgItem(hDlg,156));
    BLDExitCtrlFont(hDlg,158);
    BLDExitBrush(GetDlgItem(hDlg,158));
    BLDExitCtrlFont(hDlg,160);
    BLDExitBrush(GetDlgItem(hDlg,160));
    BLDExitCtrlFont(hDlg,162);
    BLDExitBrush(GetDlgItem(hDlg,162));
    break;

#ifdef WIN32
    case WM_CTLCOLORMSGBOX:
    case WM_CTLCOLOREDIT:
    case WM_CTLCOLORLISTBOX:
    case WM_CTLCOLORBTN:
    case WM_CTLCOLORDLG:
    case WM_CTLCOLORSCROLLBAR:
    case WM_CTLCOLORSTATIC:
#else
    case WM_CTLCOLOR:
#endif
    // Extracting data from message
    {
        HWND          hCtrl;

#ifdef WIN32
        hCtrl          = (HWND)lParam;
#else
        hCtrl          = (HWND)LOWORD(lParam);
#endif
    }

#ifdef WIN32
    if (message == WM_CTLCOLORDLG)
        return (BOOL)BLDctlColorBrushSetOrg(hDlg,(HDC)wParam);
#else
    if (HIWORD(lParam) == CTLCOLOR_DLG)
        return (BOOL)BLDctlColorBrushSetOrg(hDlg,(HDC)wParam);
#endif

    switch (GetDlgCtrlID(hCtrl))
    {
```

```
case 147:
    SetTextColor((HDC)wParam, RGB(0,0,0));
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 149:
    SetTextColor((HDC)wParam, RGB(0,0,0));
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 132:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 140:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 136:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 141:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 143:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 133:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 134:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 135:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 139:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 137:
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 150:
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 151:
```

```
        SetBkMode( (HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 152:
        SetBkMode( (HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 153:
        SetBkMode( (HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 154:
        SetBkMode( (HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 155:
        SetBkMode( (HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 156:
        SetBkColor( (HDC)wParam, RGB(255,255,255));
        SetBkMode( (HDC)wParam, OPAQUE);
        return (BOOL)BLDctlColorStockBrush(hCtrl, WHITE_BRUSH);
        break;
    case 158:
        SetBkMode( (HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 160:
        SetTextColor( (HDC)wParam, RGB(0,0,128));
        SetBkMode( (HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 162:
        SetTextColor( (HDC)wParam, RGB(0,0,128));
        SetBkMode( (HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    }
    break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//          Modal Dialog Box: MAIN1
// *****

// Startup procedure for modal dialog box
int BLD_FunctionDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_FunctionDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"MAIN1"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue== -1)
```



```
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"MAIN1"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_FunctionDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    BOOL      bRet;

    bRet      = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        BLDInitSolidBrush(hDlg,RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg,116),RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg,106),RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg,123),RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg,124),RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg,121),RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg,114),RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg,117),RGB(192,192,192));
        break;

    case WM_COMMAND:
        {
            HWND      hCtrl;

            // Extracting data from message
            hCtrl      = (HWND)(UINT)lParam;
            if(!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand(hDlg,message,wParam,lParam))
                    return TRUE;
            }
        }
        break;

    case WM_DRAWITEM:
        {
            LPDRAWITEMSTRUCT lpDrawItem;

            lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
            switch(lpDrawItem->CtlID)
            {
            case 100:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"WELCOME",TRUE);
                return TRUE;
                break;
            case 115:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"ARCHES",TRUE);
                return TRUE;
                break;
            }
        }
    }
}
```

```
        default:
            if (BLDDrawItem(hDlg, lpDrawItem))
                return TRUE;
            break;
        }
    break;

case WM_DESTROY:
    BLDExitBrush(hDlg);
    BLDExitBrush(GetDlgItem(hDlg, 116));
    BLDExitBrush(GetDlgItem(hDlg, 106));
    BLDExitBrush(GetDlgItem(hDlg, 123));
    BLDExitBrush(GetDlgItem(hDlg, 124));
    BLDExitBrush(GetDlgItem(hDlg, 121));
    BLDExitBrush(GetDlgItem(hDlg, 114));
    BLDExitBrush(GetDlgItem(hDlg, 117));
    break;

#ifdef WIN32
case WM_CTLCOLORMSGBOX:
case WM_CTLCOLOREDIT:
case WM_CTLCOLORLISTBOX:
case WM_CTLCOLORBTN:
case WM_CTLCOLORDLG:
case WM_CTLCOLORSCROLLBAR:
case WM_CTLCOLORSTATIC:
#else
case WM_CTLCOLOR:
#endif
    // Extracting data from message
    {
        HWND          hCtrl;

#ifdef WIN32
        hCtrl          = (HWND) lParam;
#else
        hCtrl          = (HWND) LOWORD(lParam);
#endif
#ifdef WIN32
        if (message == WM_CTLCOLORDLG)
            return (BOOL) BLDctlColorBrushSetOrg(hDlg, (HDC) wParam);
#else
        if (HIWORD(lParam) == CTLCOLOR_DLG)
            return (BOOL) BLDctlColorBrushSetOrg(hDlg, (HDC) wParam);
#endif
#ifdef WIN32
        switch (GetDlgCtrlID(hCtrl))
        {
            case 116:
                SetBkMode((HDC) wParam, TRANSPARENT);
                return (BOOL) BLDctlColorPropBrush(hCtrl);
                break;
            case 106:
                SetBkMode((HDC) wParam, TRANSPARENT);
                return (BOOL) BLDctlColorPropBrush(hCtrl);
                break;
            case 123:
                SetBkMode((HDC) wParam, TRANSPARENT);
                return (BOOL) BLDctlColorPropBrush(hCtrl);
                break;
            case 124:
                SetBkMode((HDC) wParam, TRANSPARENT);
                return (BOOL) BLDctlColorPropBrush(hCtrl);
                break;
        }
    }
}
```

```
        case 121:
            SetBkMode((HDC)wParam, TRANSPARENT);
            return (BOOL)BLDctlColorPropBrush(hCtrl);
            break;
        case 114:
            SetBkMode((HDC)wParam, TRANSPARENT);
            return (BOOL)BLDctlColorPropBrush(hCtrl);
            break;
        case 117:
            SetBkMode((HDC)wParam, TRANSPARENT);
            return (BOOL)BLDctlColorPropBrush(hCtrl);
            break;
        }
    }
    break;

default:
    break;
}
return bRet;                // No explicit return - return default
}

// *****
//                               Modal Dialog Box: EXIT
// *****

// Startup procedure for modal dialog box
int BLD_Function2DlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC    lpProc;
    int         ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_Function2DlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"EXIT"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"EXIT"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_Function2DlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {
        case WM_INITDIALOG:
            bRet        = TRUE;    // Default return for WM_INITDIALOG is TRUE
            BLDInitSolidBrush(hDlg, RGB(224, 223, 225));
            BLDInitSolidBrush(GetDlgItem(hDlg, 100), RGB(224, 223, 225));
            BLDInitCtrlFont(hDlg, 100, -16, 0, 0, 0, 700, 0, 0, 0, 0, 3, 2, 1, 34, "Arial");
            break;
    }
}
```



```
case WM_COMMAND:
{
    WORD        wId;
    WORD        notification;
    HWND        hCtrl;

    // Extracting data from message
    wId          = LOWORD(wParam);
    hCtrl        = (HWND) (UINT) lParam;

#ifdef WIN32
    notification = HIWORD(wParam);
#else
    notification = HIWORD(lParam);
#endif

    if(!hCtrl)                // Menu input or CR
    {
        if (BLDMenuCommand(hDlg,message,wParam,lParam))
            return TRUE;
    }

    switch(wId)
    {
    case 1:
        switch(notification)
        {
        case BN_CLICKED:
            BLD_QuitFuncUDCFunc(hDlg,message,wParam,lParam);
            EndDialog(hDlg,1);
            return TRUE;
            break;
        default:
            break;
        }
        break;
    case 2:
        switch(notification)
        {
        case BN_CLICKED:
            EndDialog(hDlg,2);
            return TRUE;
            break;
        default:
            break;
        }
        break;
    default:
        break;
    }
    break;

case WM_DRAWITEM:
{
    LPDRAWITEMSTRUCT lpDrawItem;

    lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
    switch(lpDrawItem->CtlID)
    {
    case 102:
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDrawIcon(lpDrawItem,"EXIT2");
        return TRUE;
        break;
    case 104:
```

```
        if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
            if (lpDrawItem->CtlType==ODT_BUTTON)
                BLDDrawBitmap(lpDrawItem,"X",TRUE);
        return TRUE;
        break;
    default:
        if (BLDDrawItem(hDlg,lpDrawItem))
            return TRUE;
        break;
    }
    break;

case WM_DESTROY:
    BLDExitBrush(hDlg);
    BLDExitCtrlFont(hDlg,100);
    BLDExitBrush(GetDlgItem(hDlg,100));
    break;

#ifdef WIN32
    case WM_CTLCOLORMSGBOX:
    case WM_CTLCOLOREDIT:
    case WM_CTLCOLORLISTBOX:
    case WM_CTLCOLORBTN:
    case WM_CTLCOLORDLG:
    case WM_CTLCOLORSCROLLBAR:
    case WM_CTLCOLORSTATIC:
#else
    case WM_CTLCOLOR:
#endif
    // Extracting data from message
    {
        HWND          hCtrl;

#ifdef WIN32
        hCtrl          = (HWND)lParam;
#else
        hCtrl          = (HWND)LOWORD(lParam);
#endif
#ifdef WIN32
        if(message == WM_CTLCOLORDLG)
            return (BOOL)BLDctlColorBrushSetOrg(hDlg,(HDC)wParam);
#else
        if(HIWORD(lParam) == CTLCOLOR_DLG)
            return (BOOL)BLDctlColorBrushSetOrg(hDlg,(HDC)wParam);
#endif
#ifdef WIN32
        switch(GetDlgCtrlID(hCtrl))
        {
            case 100:
                SetBkColor((HDC)wParam,RGB(192,192,192));
                SetBkMode((HDC)wParam,TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
        }
    }
    break;

default:
    break;
}
return bRet;                                // No explicit return - return default
}
```

```
// *****
//                               Modal Dialog Box: BINFO
// *****

// Startup procedure for modal dialog box
int BLD_Function6DlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_Function6DlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"BINFO"),
                          hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"BINFO"),
                          MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_Function6DlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL          bRet;

    bRet          = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet          = TRUE;    // Default return for WM_INITDIALOG is TRUE
        BLDInitSolidBrush(hDlg, RGB(224, 223, 225));
        BLDInitSolidBrush(GetDlgItem(hDlg, 147), RGB(192, 192, 192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 149), RGB(192, 192, 192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 137), RGB(192, 192, 192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 150), RGB(192, 192, 192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 151), RGB(192, 192, 192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 152), RGB(192, 192, 192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 153), RGB(192, 192, 192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 154), RGB(192, 192, 192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 155), RGB(192, 192, 192));
        break;

    case WM_COMMAND:
        {
            WORD      wId;
            WORD      notification;
            HWND      hCtrl;

            // Extracting data from message
            wId        = LOWORD(wParam);
            hCtrl       = (HWND)(UINT)lParam;

#ifdef WIN32
            notification = HIWORD(wParam);
#else
            notification = HIWORD(lParam);
#endif

            if (!hCtrl)
                // Menu input or CR
            {

```



```
    if (BLDMenuCommand(hDlg,message,wParam,lParam))
        return TRUE;
    }
    switch(wId)
    {
    case 124:
        switch(notification)
        {
        case BN_CLICKED:
            BLD_HeaderDetailDlgFunc(hDlg,message,wParam,lParam);
            return TRUE;
            break;
        default:
            break;
        }
        break;
    case ID_BRepTot:
        switch(notification)
        {
        case BN_CLICKED:
            BLD_ReportTotalsDlgFunc(hDlg,message,wParam,lParam);
            return TRUE;
            break;
        default:
            break;
        }
        break;
    case IDPRINTB:
        switch(notification)
        {
        case BN_CLICKED:
            BLD_PrintBlanketDlgFunc(hDlg,message,wParam,lParam);
            return TRUE;
            break;
        default:
            break;
        }
        break;
    default:
        break;
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        if (BLDDrawItem(hDlg,lpDrawItem))
            return TRUE;
    }
    break;

case WM_DESTROY:
    BLDExitBrush(hDlg);
    BLDExitBrush(GetDlgItem(hDlg,147));
    BLDExitBrush(GetDlgItem(hDlg,149));
    BLDExitBrush(GetDlgItem(hDlg,137));
    BLDExitBrush(GetDlgItem(hDlg,150));
    BLDExitBrush(GetDlgItem(hDlg,151));
    BLDExitBrush(GetDlgItem(hDlg,152));
    BLDExitBrush(GetDlgItem(hDlg,153));
```

```
BLDExitBrush(GetDlgItem(hDlg,154));
BLDExitBrush(GetDlgItem(hDlg,155));
BLDExitBrush(GetDlgItem(hDlg,156));
break;
```

```
#ifdef WIN32
```

```
case WM_CTLCOLORMSGBOX:
case WM_CTLCOLOREDIT:
case WM_CTLCOLORLISTBOX:
case WM_CTLCOLORBTN:
case WM_CTLCOLORDLG:
case WM_CTLCOLORSCROLLBAR:
case WM_CTLCOLORSTATIC:
```

```
#else
```

```
case WM_CTLCOLOR:
```

```
#endif
```

```
// Extracting data from message
```

```
{
    HWND      hCtrl;
```

```
#ifdef WIN32
```

```
hCtrl      = (HWND)lParam;
```

```
#else
```

```
hCtrl      = (HWND)LOWORD(lParam);
```

```
#endif
```

```
#ifdef WIN32
```

```
if(message == WM_CTLCOLORDLG)
    return (BOOL)BLDctlColorBrushSetOrg(hDlg, (HDC)wParam);
```

```
#else
```

```
if(HIWORD(lParam) == CTLCOLOR_DLG)
    return (BOOL)BLDctlColorBrushSetOrg(hDlg, (HDC)wParam);
```

```
#endif
```

```
switch(GetDlgCtrlID(hCtrl))
```

```
{
```

```
case 147:
```

```
SetTextColor((HDC)wParam, RGB(0,0,0));
SetBkMode((HDC)wParam, TRANSPARENT);
return (BOOL)BLDctlColorPropBrush(hCtrl);
break;
```

```
case 149:
```

```
SetTextColor((HDC)wParam, RGB(0,0,0));
SetBkMode((HDC)wParam, TRANSPARENT);
return (BOOL)BLDctlColorPropBrush(hCtrl);
break;
```

```
case 132:
```

```
SetTextColor((HDC)wParam, RGB(0,0,128));
SetBkMode((HDC)wParam, OPAQUE);
return (BOOL)BLDctlColorDefaultBrush(hCtrl);
break;
```

```
case 140:
```

```
SetTextColor((HDC)wParam, RGB(0,0,128));
SetBkMode((HDC)wParam, OPAQUE);
return (BOOL)BLDctlColorDefaultBrush(hCtrl);
break;
```

```
case 136:
```

```
SetTextColor((HDC)wParam, RGB(0,0,128));
SetBkMode((HDC)wParam, OPAQUE);
return (BOOL)BLDctlColorDefaultBrush(hCtrl);
break;
```

```
case 141:
```

```
SetTextColor((HDC)wParam, RGB(0,0,128));
SetBkMode((HDC)wParam, OPAQUE);
return (BOOL)BLDctlColorDefaultBrush(hCtrl);
break;
```

```
case 143:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 133:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 134:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 135:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 139:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 137:
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 150:
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 151:
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 152:
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 153:
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 154:
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 155:
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 156:
    SetBkColor((HDC)wParam, RGB(255,255,255));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorStockBrush(hCtrl, WHITE_BRUSH);
    break;
}
break;

default:
    break;
```



```
    }
    return bRet;          // No explicit return - return default
}

// *****
//           Modal Dialog Box: HDETAIL
// *****

// Startup procedure for modal dialog box
int BLD_HeaderDetailDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_HeaderDetailDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"HDETAIL"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"HDETAIL"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_HeaderDetailDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {
        case WM_INITDIALOG:
            bRet        = TRUE;    // Default return for WM_INITDIALOG is TRUE
            break;

        case WM_COMMAND:
            {
                WORD        wId;
                WORD        notification;
                HWND        hCtrl;

                // Extracting data from message
                wId        = LOWORD(wParam);
                hCtrl      = (HWND)(UINT)lParam;

#ifdef WIN32
                notification = HIWORD(wParam);
#else
                notification = HIWORD(lParam);
#endif

                if (!hCtrl)    // Menu input or CR
                {
                    if (BLDMenuCommand(hDlg, message, wParam, lParam))
                        return TRUE;
                }
                switch(wId)
                {

```



```
// *****
```

```
// Startup procedure for modal dialog box
```

```
int BLD_EmployeeDetailDlgFuncDef(HWND hWnd, char *szDlgName)
```

```
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_EmployeeDetailDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"EDETAIL"),
                          hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"EDETAIL"),
                          MB_OK | MB_ICONHAND);
    return ReturnValue;
}
```

```
// Default dialog box procedure
```

```
BOOL BLD_EmployeeDetailDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
```

```
{
    BOOL          bRet;

    bRet          = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        break;

    case WM_COMMAND:
        {
            WORD      wId;
            WORD      notification;
            HWND      hCtrl;

            // Extracting data from message
            wId      = LOWORD(wParam);
            hCtrl    = (HWND)(UINT)lParam;

#ifdef WIN32
            notification = HIWORD(wParam);
#else
            notification = HIWORD(lParam);
#endif

            if (!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand(hDlg, message, wParam, lParam))
                    return TRUE;
            }

            switch(wId)
            {
            case 101:
                switch(notification)
                {
                case BN_CLICKED:
                    EndDialog(hDlg, 101);
                    return TRUE;
                break;
                }
            }
        }
    }
}
```



```

        default:
            break;
        }
        break;
    default:
        break;
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        switch(lpDrawItem->CtlID)
        {
            case 104:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDDrawIcon(lpDrawItem,"FFIND");
                return TRUE;
                break;
            case 106:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDDrawIcon(lpDrawItem,"FFIND");
                return TRUE;
                break;
            default:
                if (BLDDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

default:
    break;
}
return bRet;                // No explicit return - return default
}

// *****
//                               Modal Dialog Box: BRREPORT
// *****

//  Startup procedure for modal dialog box
int BLD_BrowseReportDlgFuncDef(HWND hWnd,char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_BrowseReportDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"BRREPORT"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue==-1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"BRREPORT"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

```

```
// Default dialog box procedure
```

```
BOOL BLD_BrowseReportDlgDefault(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
```

```
{
```

```
    BOOL        bRet;
```

```
    bRet        = FALSE;    // Default return value if not processed
```

```
    if (BLDDlgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))  
        return bRet;
```

```
    switch(message)
```

```
    {
```

```
    case WM_INITDIALOG:
```

```
        bRet        = TRUE;    // Default return for WM_INITDIALOG is TRUE
```

```
        BLDInitSolidBrush(hDlg,RGB(224,223,225));
```

```
        BLDInitSolidBrush(GetDlgItem(hDlg,147),RGB(192,192,192));
```

```
        BLDInitSolidBrush(GetDlgItem(hDlg,149),RGB(192,192,192));
```

```
        BLDInitSolidBrush(GetDlgItem(hDlg,137),RGB(192,192,192));
```

```
        BLDInitSolidBrush(GetDlgItem(hDlg,150),RGB(192,192,192));
```

```
        BLDInitSolidBrush(GetDlgItem(hDlg,151),RGB(192,192,192));
```

```
        BLDInitSolidBrush(GetDlgItem(hDlg,152),RGB(192,192,192));
```

```
        BLDInitSolidBrush(GetDlgItem(hDlg,153),RGB(192,192,192));
```

```
        BLDInitSolidBrush(GetDlgItem(hDlg,154),RGB(192,192,192));
```

```
        BLDInitSolidBrush(GetDlgItem(hDlg,155),RGB(192,192,192));
```

```
        break;
```

```
    case WM_COMMAND:
```

```
    {
```

```
        WORD        wId;
```

```
        WORD        notification;
```

```
        HWND        hCtrl;
```

```
        // Extracting data from message
```

```
        wId          = LOWORD(wParam);
```

```
        hCtrl         = (HWND) (UINT) lParam;
```

```
#ifdef WIN32
```

```
        notification = HIWORD(wParam);
```

```
#else
```

```
        notification = HIWORD(lParam);
```

```
#endif
```

```
        if(!hCtrl)    // Menu input or CR
```

```
        {
```

```
            if (BLDMenuCommand(hDlg,message,wParam,lParam))
```

```
                return TRUE;
```

```
        }
```

```
        switch(wId)
```

```
        {
```

```
        case 124:
```

```
            switch(notification)
```

```
            {
```

```
            case BN_CLICKED:
```

```
                BLD_HeaderDetailDlgFunc(hDlg,message,wParam,lParam);
```

```
                return TRUE;
```

```
                break;
```

```
            default:
```

```
                break;
```

```
            }
```

```
            break;
```

```
        case ID_BrRepTot:
```

```
            switch(notification)
```

```
            {
```

```
            case BN_CLICKED:
```

```
        BLD_ReportTotalsDlgFunc(hDlg,message,wParam,lParam);
        return TRUE;
        break;
    default:
        break;
    }
    break;
default:
    break;
}
break;

case WM_DRAWITEM:
{
    LPDRAWITEMSTRUCT lpDrawItem;

    lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
    if(BLDDrawItem(hDlg,lpDrawItem))
        return TRUE;
}
break;
break;

case WM_DESTROY:
    BLDExitBrush(hDlg);
    BLDExitBrush(GetDlgItem(hDlg,147));
    BLDExitBrush(GetDlgItem(hDlg,149));
    BLDExitBrush(GetDlgItem(hDlg,137));
    BLDExitBrush(GetDlgItem(hDlg,150));
    BLDExitBrush(GetDlgItem(hDlg,151));
    BLDExitBrush(GetDlgItem(hDlg,152));
    BLDExitBrush(GetDlgItem(hDlg,153));
    BLDExitBrush(GetDlgItem(hDlg,154));
    BLDExitBrush(GetDlgItem(hDlg,155));
    BLDExitBrush(GetDlgItem(hDlg,156));
    break;

#ifdef WIN32
    case WM_CTLCOLORMSGBOX:
    case WM_CTLCOLOREDIT:
    case WM_CTLCOLORLISTBOX:
    case WM_CTLCOLORBTN:
    case WM_CTLCOLORDLG:
    case WM_CTLCOLORSCROLLBAR:
    case WM_CTLCOLORSTATIC:
#else
    case WM_CTLCOLOR:
#endif
    // Extracting data from message
    {
        HWND          hCtrl;

#ifdef WIN32
        hCtrl          = (HWND)lParam;
#else
        hCtrl          = (HWND)LOWORD(lParam);
#endif
    }
#ifdef WIN32
    if(message == WM_CTLCOLORDLG)
        return (BOOL)BLDCtlColorBrushSetOrg(hDlg,(HDC)wParam);
#else
    if(HIWORD(lParam) == CTLCOLOR_DLG)
        return (BOOL)BLDCtlColorBrushSetOrg(hDlg,(HDC)wParam);
#endif
}
```



#endif

```
switch(GetDlgCtrlID(hCtrl))
{
case 147:
    SetTextColor((HDC)wParam, RGB(0,0,0));
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 149:
    SetTextColor((HDC)wParam, RGB(0,0,0));
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 132:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 140:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 136:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 141:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 143:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 133:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 134:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 135:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 139:
    SetTextColor((HDC)wParam, RGB(0,0,128));
    SetBkMode((HDC)wParam, OPAQUE);
    return (BOOL)BLDctlColorDefaultBrush(hCtrl);
    break;
case 137:
    SetBkMode((HDC)wParam, TRANSPARENT);
    return (BOOL)BLDctlColorPropBrush(hCtrl);
    break;
case 150:
    SetBkMode((HDC)wParam, TRANSPARENT);
```

```

        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 151:
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 152:
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 153:
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 154:
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 155:
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 156:
        SetBkColor((HDC)wParam, RGB(255,255,255));
        SetBkMode((HDC)wParam, OPAQUE);
        return (BOOL)BLDctlColorStockBrush(hCtrl, WHITE_BRUSH);
        break;
    }
    break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//          Modal Dialog Box: STATISTICS
// *****

// Startup procedure for modal dialog box
int BLD_ReportStatisticsDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_ReportStatisticsDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"STATISTICS"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"STATISTICS"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_ReportStatisticsDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL          bRet;

```

```
bRet      = FALSE;    // Default return value if not processed

if (BLDDLgMsgFilter(hDlg,message,wParam,lParam,BLDDLGMODAL,0,&bRet))
    return bRet;

switch(message)
{

case WM_INITDIALOG:
    bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
    break;

case WM_COMMAND:
    {
        HWND      hCtrl;

        // Extracting data from message
        hCtrl      = (HWND) (UINT) lParam;
        if(!hCtrl)    // Menu input or CR
        {
            if (BLDMenuCommand(hDlg,message,wParam,lParam))
                return TRUE;
        }
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
        switch(lpDrawItem->CtlID)
        {
            case 102:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"CHART1");
                return TRUE;
                break;
            case 104:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"CHART3");
                return TRUE;
                break;
            case 105:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"CHART4");
                return TRUE;
                break;
            case 106:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawIcon(lpDrawItem,"CHART5");
                return TRUE;
                break;
            default:
                if (BLDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;
```



```

    default:
        break;
    }
    return bRet;          // No explicit return - return default
}

// *****
//                               Modal Dialog Box: REPORT
// *****

// Startup procedure for modal dialog box
int BLD_ReportDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_ReportDlgProc, hWnd);
    ReturnValue = DialogBox(hWnd, (LPSTR)(szDlgName?szDlgName:"REPORT"),
                            hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"REPORT"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// .Default dialog box procedure
BOOL BLD_ReportDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL        bRet;

    bRet = FALSE;        // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch (message)
    {
        case WM_INITDIALOG:
            bRet = TRUE;        // Default return for WM_INITDIALOG is TRUE
            BLDInitSolidBrush(GetDlgItem(hDlg, 109), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 115), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 118), RGB(192, 192, 192));
            break;

        case WM_COMMAND:
            {
                WORD        wId;
                WORD        notification;
                HWND        hCtrl;

                // Extracting data from message
                wId = LOWORD(wParam);
                hCtrl = (HWND)(UINT)lParam;

#ifdef WIN32
                notification = HIWORD(wParam);
#else
                notification = HIWORD(lParam);
#endif

                if (!hCtrl)          // Menu input or CR

```

```
    {
        if (BLDMenuCommand(hDlg,message,wParam,lParam))
            return TRUE;
    }
    switch(wId)
    {
    case 1:
        switch(notification)
        {
            case BN_CLICKED:
                BLD_ReportStatisticsDlgFunc(hDlg,message,wParam,lParam);
                EndDialog(hDlg,1);
                return TRUE;
            break;
            default:
                break;
        }
        break;
    default:
        break;
    }
}
break;

case WM_DRAWITEM:
{
    LPDRAWITEMSTRUCT lpDrawItem;

    lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
    if (BLDDrawItem(hDlg,lpDrawItem))
        return TRUE;
}
break;

case WM_DESTROY:
    BLDExitBrush(GetDlgItem(hDlg,109));
    BLDExitBrush(GetDlgItem(hDlg,115));
    BLDExitBrush(GetDlgItem(hDlg,118));
    break;

#ifdef WIN32
    case WM_CTLCOLORMSGBOX:
    case WM_CTLCOLOREDIT:
    case WM_CTLCOLORLISTBOX:
    case WM_CTLCOLORBTN:
    case WM_CTLCOLORDLG:
    case WM_CTLCOLORSCROLLBAR:
    case WM_CTLCOLORSTATIC:
#else
    case WM_CTLCOLOR:
#endif
// Extracting data from message
{
    HWND          hCtrl;

#ifdef WIN32
    hCtrl          = (HWND)lParam;
#else
    hCtrl          = (HWND)LOWORD(lParam);
#endif
    switch(GetDlgCtrlID(hCtrl))
    {
    case 109:
```

```

        SetTextColor((HDC)wParam, RGB(0,0,128));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 115:
        SetTextColor((HDC)wParam, RGB(0,0,128));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 118:
        SetTextColor((HDC)wParam, RGB(0,0,128));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    }
}
break;

default:
    break;
}
return bRet;          // No explicit return - return default
}

// *****
//                               Modal Dialog Box: NEWBROWSE
// *****

// Startup procedure for modal dialog box
int BLD_BrowseEntryDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC    lpProc;
    int         ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_BrowseEntryDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"NEWBROWSE"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue==-1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"NEWBROWSE"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_BrowseEntryDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL        bRet;

    bRet        = FALSE;    // Default return value if not processed

    if(BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {

    case WM_INITDIALOG:
        bRet        = TRUE;    // Default return for WM_INITDIALOG is TRUE
        BLDInitSolidBrush(GetDlgItem(hDlg, 109), RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 115), RGB(192,192,192));
        BLDInitSolidBrush(GetDlgItem(hDlg, 118), RGB(192,192,192));
    }
}

```



```
    BLDInitSolidBrush(GetDlgItem(hDlg,121),RGB(192,192,192));
    break;

case WM_COMMAND:
{
    WORD        wId;
    WORD        notification;
    HWND        hCtrl;

    // Extracting data from message
    wId         = LOWORD(wParam);
    hCtrl       = (HWND)(UINT)lParam;

#ifdef WIN32
    notification = HIWORD(wParam);
#else
    notification = HIWORD(lParam);
#endif

    if(!hCtrl)           // Menu input or CR
    {
        if (BLDMenuCommand(hDlg,message,wParam,lParam))
            return TRUE;
    }

    switch(wId)
    {
    case 1:
        switch(notification)
        {
            case BN_CLICKED:
                BLD_BrowseReportDlgFunc(hDlg,message,wParam,lParam);
                return TRUE;
                break;
            default:
                break;
        }
        break;
    default:
        break;
    }
}
break;

case WM_DRAWITEM:
{
    LPDRAWITEMSTRUCT lpDrawItem;

    lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
    if (BLDDrawItem(hDlg,lpDrawItem))
        return TRUE;
}
break;

case WM_DESTROY:
    BLDExitBrush(GetDlgItem(hDlg,109));
    BLDExitBrush(GetDlgItem(hDlg,115));
    BLDExitBrush(GetDlgItem(hDlg,118));
    BLDExitBrush(GetDlgItem(hDlg,121));
    break;

#ifdef WIN32
case WM_CTLCOLORMSGBOX:
case WM_CTLCOLOREDIT:
case WM_CTLCOLORLISTBOX:
case WM_CTLCOLORBTN:
```

```

        case WM_CTLCOLORDLG:
        case WM_CTLCOLORSCROLLBAR:
        case WM_CTLCOLORSTATIC:
    #else
        case WM_CTLCOLOR:
    #endif
        // Extracting data from message
        {
            HWND          hCtrl;

#ifdef WIN32
            hCtrl          = (HWND)lParam;
#else
            hCtrl          = (HWND)LOWORD(lParam);
#endif
            switch(GetDlgCtrlID(hCtrl))
            {
                case 109:
                    SetTextColor((HDC)wParam, RGB(0,0,128));
                    SetBkMode((HDC)wParam, TRANSPARENT);
                    return (BOOL)BLDctlColorPropBrush(hCtrl);
                    break;
                case 115:
                    SetTextColor((HDC)wParam, RGB(0,0,128));
                    SetBkMode((HDC)wParam, TRANSPARENT);
                    return (BOOL)BLDctlColorPropBrush(hCtrl);
                    break;
                case 118:
                    SetTextColor((HDC)wParam, RGB(0,0,128));
                    SetBkMode((HDC)wParam, TRANSPARENT);
                    return (BOOL)BLDctlColorPropBrush(hCtrl);
                    break;
                case 121:
                    SetTextColor((HDC)wParam, RGB(0,0,128));
                    SetBkMode((HDC)wParam, TRANSPARENT);
                    return (BOOL)BLDctlColorPropBrush(hCtrl);
                    break;
            }
            break;

        default:
            break;
        }
        return bRet;                // No explicit return - return default
    }

// *****
//                               Modal Dialog Box: TOTALS
// *****

// Startup procedure for modal dialog box
int BLD_ReportTotalsDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_ReportTotalsDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"TOTALS"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"TOTALS"),

```

```
                                MB_OK | MB_ICONHAND);
return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_ReportTotalsDlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL          bRet;

    bRet          = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter (hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch (message)
    {

    case WM_INITDIALOG:
        bRet          = TRUE;    // Default return for WM_INITDIALOG is TRUE
        BLDInitSolidBrush (GetDlgItem (hDlg, 109), RGB (0, 0, 128));
        BLDInitSolidBrush (GetDlgItem (hDlg, 110), RGB (0, 0, 128));
        BLDInitSolidBrush (GetDlgItem (hDlg, 111), RGB (0, 0, 128));
        BLDInitSolidBrush (GetDlgItem (hDlg, 112), RGB (0, 0, 128));
        BLDInitSolidBrush (GetDlgItem (hDlg, 113), RGB (0, 0, 128));
        break;

    case WM_COMMAND:
        {
            WORD          wId;
            WORD          notification;
            HWND          hCtrl;

            // Extracting data from message
            wId          = LOWORD (wParam);
            hCtrl        = (HWND) (UINT) lParam;

#ifdef WIN32
            notification = HIWORD (wParam);
#else
            notification = HIWORD (lParam);
#endif

            if (!hCtrl)                // Menu input or CR
            {
                if (BLDMenuCommand (hDlg, message, wParam, lParam))
                    return TRUE;
            }

            switch (wId)
            {
            case IDPRINTTOT:
                switch (notification)
                {
                case BN_CLICKED:
                    EndDialog (hDlg, IDPRINTTOT);
                    return TRUE;
                    break;
                default:
                    break;
                }
                break;
            case ID_CONT:
                switch (notification)
                {
                case BN_CLICKED:
                    EndDialog (hDlg, ID_CONT);

```



```
        return TRUE;
        break;
    default:
        break;
    }
    break;
default:
    break;
}
break;

case WM_DRAWITEM:
{
    LPDRAWITEMSTRUCT lpDrawItem;

    lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
    if (BLDDrawItem(hDlg, lpDrawItem))
        return TRUE;
}
break;
break;

case WM_DESTROY:
    BLDExitBrush(GetDlgItem(hDlg, 109));
    BLDExitBrush(GetDlgItem(hDlg, 110));
    BLDExitBrush(GetDlgItem(hDlg, 111));
    BLDExitBrush(GetDlgItem(hDlg, 112));
    BLDExitBrush(GetDlgItem(hDlg, 113));
    break;

#ifdef WIN32
case WM_CTLCOLORMSGBOX:
case WM_CTLCOLOREDIT:
case WM_CTLCOLORLISTBOX:
case WM_CTLCOLORBTN:
case WM_CTLCOLORDLG:
case WM_CTLCOLORSCROLLBAR:
case WM_CTLCOLORSTATIC:
#else
case WM_CTLCOLOR:
#endif
    // Extracting data from message
    {
        HWND          hCtrl;

#ifdef WIN32
        hCtrl          = (HWND)lParam;
#else
        hCtrl          = (HWND)LOWORD(lParam);
#endif

        switch(GetDlgCtrlID(hCtrl))
        {
            case ID_TotEIN:
                SetTextColor((HDC)wParam, RGB(0, 0, 0));
                SetBkMode((HDC)wParam, OPAQUE);
                return (BOOL)BLDctlColorDefaultBrush(hCtrl);
                break;
            case 101:
                SetTextColor((HDC)wParam, RGB(0, 0, 128));
                SetBkMode((HDC)wParam, OPAQUE);
                return (BOOL)BLDctlColorDefaultBrush(hCtrl);
                break;
            case ID_TotRpt:

```

```
        SetTextColor((HDC)wParam,RGB(0,0,0));
        SetBkMode((HDC)wParam,OPAQUE);
        return (BOOL)BLDctlColorDefaultBrush(hCtrl);
        break;
    case 107:
        SetTextColor((HDC)wParam,RGB(0,0,128));
        SetBkMode((HDC)wParam,OPAQUE);
        return (BOOL)BLDctlColorDefaultBrush(hCtrl);
        break;
    case 108:
        SetTextColor((HDC)wParam,RGB(0,0,128));
        SetBkMode((HDC)wParam,OPAQUE);
        return (BOOL)BLDctlColorDefaultBrush(hCtrl);
        break;
    case 109:
        SetBkMode((HDC)wParam,TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 110:
        SetBkMode((HDC)wParam,TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 111:
        SetBkMode((HDC)wParam,TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 112:
        SetBkMode((HDC)wParam,TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 113:
        SetBkMode((HDC)wParam,TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    }
    break;

default:
    break;
}
return bRet;          // No explicit return - return default
}
```

```
// *****
//                               Modal Dialog Box: SEQ
// *****
```

```
// Startup procedure for modal dialog box
```

```
int BLD_SequenceDlgFuncDef(HWND hWnd,char *szDlgName)
```

```
{
    DLGPROC    lpProc;
    int        ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_SequenceDlgProc,hInst);
    ReturnValue = DialogBox(hInst,(LPSTR)(szDlgName?szDlgName:"SEQ"),
                           hWnd,lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,(szDlgName?szDlgName:"SEQ"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}
```

```
// Default dialog box procedure
BOOL BLD_SequenceDlgDefault (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL      bRet;

    bRet      = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter (hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch (message)
    {

    case WM_INITDIALOG:
        bRet      = TRUE;    // Default return for WM_INITDIALOG is TRUE
        BLDInitSolidBrush (hDlg, RGB (192, 192, 192));
        BLDInitSolidBrush (GetDlgItem (hDlg, 103), RGB (192, 192, 192));
        break;

    case WM_COMMAND:
        {
            HWND      hCtrl;

            // Extracting data from message
            hCtrl      = (HWND) (UINT) lParam;
            if (!hCtrl)    // Menu input or CR
            {
                if (BLDMenuCommand (hDlg, message, wParam, lParam))
                    return TRUE;
            }
        }
        break;

    case WM_DRAWITEM:
        {
            LPDRAWITEMSTRUCT lpDrawItem;

            lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
            if (BLDDrawItem (hDlg, lpDrawItem))
                return TRUE;
        }
        break;
        break;

    case WM_DESTROY:
        BLDExitBrush (hDlg);
        BLDExitBrush (GetDlgItem (hDlg, 103));
        break;

#ifdef WIN32
    case WM_CTLCOLORMSGBOX:
    case WM_CTLCOLOREDIT:
    case WM_CTLCOLORLISTBOX:
    case WM_CTLCOLORBTN:
    case WM_CTLCOLORDLG:
    case WM_CTLCOLORSCROLLBAR:
    case WM_CTLCOLORSTATIC:
#else
    case WM_CTLCOLOR:
#endif
        // Extracting data from message
        {
```



```
        HWND          hCtrl;

#ifdef WIN32
        hCtrl          = (HWND)lParam;
#else
        hCtrl          = (HWND)LOWORD(lParam);
#endif
#ifdef WIN32
        if(message == WM_CTLCLORDLG)
            return (BOOL)BLDctlColorBrushSetOrg(hDlg, (HDC)wParam);
#else
        if(HIWORD(lParam) == CTLCOLOR_DLG)
            return (BOOL)BLDctlColorBrushSetOrg(hDlg, (HDC)wParam);
#endif
        switch(GetDlgCtrlID(hCtrl))
        {
            case 103:
                SetBkMode((HDC)wParam, TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
        }
        break;

    default:
        break;
    }
    return bRet;          // No explicit return - return default
}

// *****
//          Modal Dialog Box: QPARAM
// *****

// Startup procedure for modal dialog box
int BLD_qparamDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_qparamDlgProc, hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)(szDlgName?szDlgName:"QPARAM"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"QPARAM"),
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_qparamDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL          bRet;

    bRet          = FALSE;    // Default return value if not processed

    if(BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {

```

```
case WM_INITDIALOG:
    bRet      = TRUE;          // Default return for WM_INITDIALOG is TRUE
    BLDInitSolidBrush(hDlg, RGB(0,128,128));
    BLDInitSolidBrush(GetDlgItem(hDlg,109), RGB(0,128,128));
    BLDInitCtrlFont(hDlg,109,-16,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    BLDInitSolidBrush(GetDlgItem(hDlg,115), RGB(0,128,128));
    BLDInitCtrlFont(hDlg,115,-16,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    BLDInitSolidBrush(GetDlgItem(hDlg,119), RGB(0,128,128));
    BLDInitCtrlFont(hDlg,119,-16,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    BLDInitSolidBrush(GetDlgItem(hDlg,116), RGB(0,128,128));
    BLDInitCtrlFont(hDlg,116,-16,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    BLDInitSolidBrush(GetDlgItem(hDlg,121), RGB(0,128,128));
    BLDInitCtrlFont(hDlg,121,-16,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    BLDInitSolidBrush(GetDlgItem(hDlg,117), RGB(0,128,128));
    BLDInitCtrlFont(hDlg,117,-16,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    BLDInitSolidBrush(GetDlgItem(hDlg,124), RGB(0,128,128));
    BLDInitCtrlFont(hDlg,124,-19,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    BLDInitSolidBrush(GetDlgItem(hDlg,qp_year), RGB(0,128,128));
    BLDInitCtrlFont(hDlg,qp_year,-16,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    BLDInitSolidBrush(GetDlgItem(hDlg,qp_ein), RGB(0,128,128));
    BLDInitCtrlFont(hDlg,qp_ein,-16,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    BLDInitSolidBrush(GetDlgItem(hDlg,qp_estab), RGB(0,128,128));
    BLDInitCtrlFont(hDlg,qp_estab,-16,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    BLDInitSolidBrush(GetDlgItem(hDlg,qp_lname), RGB(0,128,128));
    BLDInitCtrlFont(hDlg,qp_lname,-16,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    BLDInitSolidBrush(GetDlgItem(hDlg,qp_fname), RGB(0,128,128));
    BLDInitCtrlFont(hDlg,qp_fname,-16,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    BLDInitSolidBrush(GetDlgItem(hDlg,qp_ssn), RGB(0,128,128));
    BLDInitCtrlFont(hDlg,qp_ssn,-16,0,0,0,700,0,0,0,0,3,2,1,34,"Arial");
    break;

case WM_COMMAND:
{
    HWND      hCtrl;

    // Extracting data from message
    hCtrl      = (HWND) (UINT) lParam;
    if(!hCtrl)          // Menu input or CR
    {
        if (BLDMenuCommand(hDlg,message,wParam,lParam))
            return TRUE;
    }
}
break;

case WM_DRAWITEM:
{
    LPDRAWITEMSTRUCT lpDrawItem;

    lpDrawItem = (LPDRAWITEMSTRUCT) lParam;
    if(BLDDrawItem(hDlg,lpDrawItem))
        return TRUE;
}
break;
break;

case WM_DESTROY:
    BLDExitBrush(hDlg);
    BLDExitCtrlFont(hDlg,109);
    BLDExitBrush(GetDlgItem(hDlg,109));
    BLDExitCtrlFont(hDlg,115);
    BLDExitBrush(GetDlgItem(hDlg,115));
    BLDExitCtrlFont(hDlg,119);
```

```
BLDExitBrush(GetDlgItem(hDlg,119));
BLDExitCtrlFont(hDlg,116);
BLDExitBrush(GetDlgItem(hDlg,116));
BLDExitCtrlFont(hDlg,121);
BLDExitBrush(GetDlgItem(hDlg,121));
BLDExitCtrlFont(hDlg,117);
BLDExitBrush(GetDlgItem(hDlg,117));
BLDExitCtrlFont(hDlg,124);
BLDExitBrush(GetDlgItem(hDlg,124));
BLDExitCtrlFont(hDlg,qp_year);
BLDExitBrush(GetDlgItem(hDlg,qp_year));
BLDExitCtrlFont(hDlg,qp_ein);
BLDExitBrush(GetDlgItem(hDlg,qp_ein));
BLDExitCtrlFont(hDlg,qp_estab);
BLDExitBrush(GetDlgItem(hDlg,qp_estab));
BLDExitCtrlFont(hDlg,qp_lname);
BLDExitBrush(GetDlgItem(hDlg,qp_lname));
BLDExitCtrlFont(hDlg,qp_fname);
BLDExitBrush(GetDlgItem(hDlg,qp_fname));
BLDExitCtrlFont(hDlg,qp_ssn);
BLDExitBrush(GetDlgItem(hDlg,qp_ssn));
break;
```

```
#ifndef WIN32
```

```
case WM_CTLCOLORMSGBOX:
case WM_CTLCOLOREDIT:
case WM_CTLCOLORLISTBOX:
case WM_CTLCOLORBTN:
case WM_CTLCOLORDLG:
case WM_CTLCOLORSCROLLBAR:
case WM_CTLCOLORSTATIC:
```

```
#else
```

```
case WM_CTLCOLOR:
```

```
#endif
```

```
// Extracting data from message
```

```
{
    HWND          hCtrl;
```

```
#ifndef WIN32
```

```
hCtrl          = (HWND)lParam;
```

```
#else
```

```
hCtrl          = (HWND)LOWORD(lParam);
```

```
#endif
```

```
#ifndef WIN32
```

```
if(message == WM_CTLCOLORDLG)
```

```
    return (BOOL)BLDctlColorBrushSetOrg(hDlg, (HDC)wParam);
```

```
#else
```

```
if(HIWORD(lParam) == CTLCOLOR_DLG)
```

```
    return (BOOL)BLDctlColorBrushSetOrg(hDlg, (HDC)wParam);
```

```
#endif
```

```
switch(GetDlgItemID(hCtrl))
```

```
{
```

```
case 109:
```

```
    SetTextColor((HDC)wParam, RGB(0,0,0));
```

```
    SetBkMode((HDC)wParam, TRANSPARENT);
```

```
    return (BOOL)BLDctlColorPropBrush(hCtrl);
```

```
    break;
```

```
case 115:
```

```
    SetTextColor((HDC)wParam, RGB(0,0,0));
```

```
    SetBkMode((HDC)wParam, TRANSPARENT);
```

```
    return (BOOL)BLDctlColorPropBrush(hCtrl);
```

```
    break;
```

```
case 119:
```

```
    SetTextColor((HDC)wParam, RGB(0,0,0));
```



```
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 116:
        SetTextColor((HDC)wParam, RGB(0,0,0));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 121:
        SetTextColor((HDC)wParam, RGB(0,0,0));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 117:
        SetTextColor((HDC)wParam, RGB(0,0,0));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 124:
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case qp_year:
        SetTextColor((HDC)wParam, RGB(255,255,255));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case qp_ein:
        SetTextColor((HDC)wParam, RGB(255,255,255));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case qp_estab:
        SetTextColor((HDC)wParam, RGB(255,255,255));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case qp_lname:
        SetTextColor((HDC)wParam, RGB(255,255,255));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case qp_fname:
        SetTextColor((HDC)wParam, RGB(255,255,255));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case qp_ssn:
        SetTextColor((HDC)wParam, RGB(255,255,255));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    }
    break;

default:
    break;
}
return bRet;                // No explicit return - return default
}
```

```
// *****
```

```
//          Modal Dialog Box: QUESTION
// *****

// Startup procedure for modal dialog box
int BLD_questDlgFuncDef(HWND hWnd, char *szDlgName)
{
    DLGPROC      lpProc;
    int          ReturnValue;

    lpProc = (DLGPROC)MakeProcInstance((FARPROC)BLD_questDlgProc, hWnd);
    ReturnValue = DialogBox(hWnd, (LPSTR)(szDlgName?szDlgName:"QUESTION"),
                           hWnd, lpProc);
    FreeProcInstance((FARPROC)lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage(hWnd, BLD_CannotCreate, (szDlgName?szDlgName:"QUESTION"),
                          MB_OK | MB_ICONHAND);

    return ReturnValue;
}

// Default dialog box procedure
BOOL BLD_questDlgDefault(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL          bRet;

    bRet          = FALSE;    // Default return value if not processed

    if (BLDDlgMsgFilter(hDlg, message, wParam, lParam, BLDDLGMODAL, 0, &bRet))
        return bRet;

    switch(message)
    {
        case WM_INITDIALOG:
            bRet = TRUE;    // Default return for WM_INITDIALOG is TRUE
            BLDInitSolidBrush(hDlg, RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 101), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 102), RGB(192, 192, 192));
            BLDInitCtrlFont(hDlg, 102, -19, 0, 0, 0, 700, 0, 0, 0, 0, 3, 2, 1, 34, "Arial");
            BLDInitSolidBrush(GetDlgItem(hDlg, 103), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 104), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, ID_Q1Yes), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, ID_Q1No), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, ID_Q2Yes), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, ID_Q2No), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 111), RGB(192, 192, 192));
            BLDInitCtrlFont(hDlg, 111, -11, 0, 0, 0, 700, 0, 0, 0, 0, 3, 2, 1, 34, "Arial");
            BLDInitSolidBrush(GetDlgItem(hDlg, 113), RGB(192, 192, 192));
            BLDInitCtrlFont(hDlg, 113, -11, 0, 0, 0, 700, 0, 0, 0, 0, 3, 2, 1, 34, "Arial");
            BLDInitSolidBrush(GetDlgItem(hDlg, 114), RGB(192, 192, 192));
            BLDInitSolidBrush(GetDlgItem(hDlg, 115), RGB(192, 192, 192));
            break;

        case WM_COMMAND:
            {
                WORD      wId;
                WORD      notification;
                HWND      hCtrl;

                // Extracting data from message
                wId        = LOWORD(wParam);
                hCtrl      = (HWND)(UINT)lParam;

#ifdef WIN32
                notification = HIWORD(wParam);

```

```
#else
    notification = HIWORD(lParam);
#endif
    if(!hCtrl)           // Menu input or CR
    {
        if (BLDMenuCommand(hDlg,message,wParam,lParam))
            return TRUE;
    }
    switch(wId)
    {
    case 1:
        switch(notification)
        {
            case BN_CLICKED:
                EndDialog(hDlg,1);
                return TRUE;
                break;
            default:
                break;
        }
        break;
    default:
        break;
    }
    break;

case WM_DRAWITEM:
    {
        LPDRAWITEMSTRUCT lpDrawItem;

        lpDrawItem = (LPDRAWITEMSTRUCT)lParam;
        switch(lpDrawItem->CtlID)
        {
            case 100:
                if (lpDrawItem->itemAction==ODA_DRAWENTIRE)
                    if (lpDrawItem->CtlType==ODT_BUTTON)
                        BLDDrawBitmap(lpDrawItem,"ENTERPRZ",TRUE);
                return TRUE;
                break;
            default:
                if (BLDDrawItem(hDlg,lpDrawItem))
                    return TRUE;
                break;
        }
    }
    break;

case WM_DESTROY:
    BLDExitBrush(hDlg);
    BLDExitBrush(GetDlgItem(hDlg,101));
    BLDExitCtrlFont(hDlg,102);
    BLDExitBrush(GetDlgItem(hDlg,102));
    BLDExitBrush(GetDlgItem(hDlg,103));
    BLDExitBrush(GetDlgItem(hDlg,104));
    BLDExitBrush(GetDlgItem(hDlg,ID_Q1Yes));
    BLDExitBrush(GetDlgItem(hDlg,ID_Q1No));
    BLDExitBrush(GetDlgItem(hDlg,ID_Q2Yes));
    BLDExitBrush(GetDlgItem(hDlg,ID_Q2No));
    BLDExitCtrlFont(hDlg,111);
    BLDExitBrush(GetDlgItem(hDlg,111));
    BLDExitCtrlFont(hDlg,113);
    BLDExitBrush(GetDlgItem(hDlg,113));
    BLDExitBrush(GetDlgItem(hDlg,114));
```



```
        BLDExitBrush(GetDlgItem(hDlg, 115));
        break;

#ifdef WIN32
    case WM_CTLCOLORMSGBOX:
    case WM_CTLCOLOREDIT:
    case WM_CTLCOLORLISTBOX:
    case WM_CTLCOLORBTN:
    case WM_CTLCOLORDLG:
    case WM_CTLCOLORSCROLLBAR:
    case WM_CTLCOLORSTATIC:
#else
    case WM_CTLCOLOR:
#endif
    // Extracting data from message
    {
        HWND          hCtrl;

#ifdef WIN32
        hCtrl          = (HWND)lParam;
#else
        hCtrl          = (HWND)LOWORD(lParam);
#endif
#ifdef WIN32
        if(message == WM_CTLCOLORDLG)
            return (BOOL)BLDctlColorBrushSetOrg(hDlg, (HDC)wParam);
#else
        if(HIWORD(lParam) == CTLCOLOR_DLG)
            return (BOOL)BLDctlColorBrushSetOrg(hDlg, (HDC)wParam);
#endif
        switch(GetDlgCtrlID(hCtrl))
        {
            case 101:
                SetTextColor((HDC)wParam, RGB(0, 0, 255));
                SetBkMode((HDC)wParam, TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case 102:
                SetTextColor((HDC)wParam, RGB(0, 0, 128));
                SetBkMode((HDC)wParam, TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case 103:
                SetBkMode((HDC)wParam, TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case 104:
                SetBkMode((HDC)wParam, TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case ID_Q1Yes:
                SetBkMode((HDC)wParam, TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case ID_Q1No:
                SetBkMode((HDC)wParam, TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case ID_Q2Yes:
                SetBkMode((HDC)wParam, TRANSPARENT);
                return (BOOL)BLDctlColorPropBrush(hCtrl);
                break;
            case ID_Q2No:
                SetBkMode((HDC)wParam, TRANSPARENT);
```

```
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 111:
        SetTextColor((HDC)wParam, RGB(0,0,0));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 113:
        SetTextColor((HDC)wParam, RGB(0,0,0));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 114:
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    case 115:
        SetTextColor((HDC)wParam, RGB(192,192,192));
        SetBkMode((HDC)wParam, TRANSPARENT);
        return (BOOL)BLDctlColorPropBrush(hCtrl);
        break;
    }
    break;

default:
    break;
}
return bRet;           // No explicit return - return default
}
```





```
//astat.c
//created 12/3/93 by Laura Downey, Computer Scientist, NIST
//utility program to convert binary stats found in file stat*.fil
//to ascii format and to write the stats to stat*.txt
//compiled using MSC6.0 pwb

//NOTE: astat.exe needs to be located in the same directory as stat#.fil

#include <stdio.h>           //standard i/o
#include <conio.h>           //console and port i/o
#include <time.h>            //time definitions and structures
#include "struct.h"         //custom structure definitions including ex_stat

struct ex_stat aggregate;   //initializes to all zeroes when defined globally

main()
{
    char c_usernum[3];       //converted user number
    char ASTATFILE[15];     //text file of statistics
    char STATFILE[15];      //holds file name entered by user
    char UFILE[15];         //holds file name of user number file

    FILE *asf, *sf, *uf;    //file pointer to pstat file, stat file
                           //& usernum file

    int num;                //was anything read from the file
    int user;               //user number read from usernum.fil
    time_t t;               //current date and time

    //get the current time and ask user for name of stat file
    t = time(NULL);
    system("cls");
    printf("PLEASE ENTER THE NAME OF THE BINARY FILE CONTAINING THE STATISTICS\n\n");
    scanf("%s", STATFILE);

    //read in the latest aggregate statistics from stat#.fil
    sf = fopen(STATFILE, "rb");
    if (sf == NULL)
    {
        printf("UNABLE TO OPEN %s, PROGRAM EXITING", STATFILE);
        exit(0);
    }
    else //sf != NULL
    {
        num = fread(&aggregate, sizeof(struct ex_stat), 1, sf);
        if (num == 0)
        {
            printf("%s IS EMPTY, PROGRAM EXITING", STATFILE);
            fclose(sf);
            exit(0);
        }
        else //get the user number
        {
            fclose(sf);
            strcpy(UFILE, "usernum.fil");
            uf = fopen(UFILE, "rt");
            if (uf == NULL)
            {
                printf("UNABLE TO OPEN %s\n, PROGRAM EXITING", UFILE);
                exit(0);
            }
            else //uf != NULL
```

```
{
num = fscanf(uf, "%d", &user);
if (num == 0)
{
printf("USERNUM.FIL WAS EMPTY, PROGRAM EXITING");
fclose(uf);
exit(0);
}
else //display and write stats to text file
{
fclose(uf);
strcpy(ASTATFILE, "stat");
strcat(ASTATFILE, itoa(user, c_usernum, 10) );
strcat(ASTATFILE, ".txt");

asf = fopen(ASTATFILE, "wt");
if (asf == NULL)
{
printf("UNABLE TO OPEN %s, PROGRAM EXITING", ASTATFILE);
exit(0);
}
else //display and write
{
system("cls");
fprintf(asf, "Today's date and time: %s\n\n", ctime(&t));

fprintf(asf, "\nTOTALS USED FOR CALCULATING AVERAGES AND PERCENTAGES\n\n");

fprintf(asf, "RUNNING TOTAL OF BROWSE REPORT TIME = %.2f\n", aggregate.br_tot_t
ime);

fprintf(asf, "RUNNING TOTAL OF SQ TIME, RES. W/NO INT = %.2f\n", aggregate.rs_t
ot_time);

fprintf(asf, "RUNNING TOTAL OF SQ TIME, UNRES. W/NO INT = %.2f\n", aggregate.us
_tot_time);

fprintf(asf, "RUNNING TOTAL OF SQ TIME, RES. W/ INT = %.2f\n", aggregate.rsi_to
t_time);

fprintf(asf, "RUNNING TOTAL OF SQ TIME, UNRES. W/INT = %.2f\n", aggregate.usi_t
ot_time);

fprintf(asf, "RUNNING TOTAL OF SQ BROWSE TIME, RES. W/NO INT = %.2f\n", aggrega
te.rq_tot_time);

fprintf(asf, "RUNNING TOTAL OF SQ BROWSE TIME, UNRES. W/NO INT = %.2f\n", aggrega
te.uq_tot_time);

fprintf(asf, "RUNNING TOTAL OF SQ BROWSE TIME, RES. W/ INT = %.2f\n", aggregate
.rqi_tot_time);

fprintf(asf, "RUNNING TOTAL OF SQ BROWSE TIME, UNRES. W/INT = %.2f\n", aggregat
e.uqi_tot_time);

fprintf(asf, "TOTAL BROWSE CASES = %ld\n", aggregate.tot_browse);

fprintf(asf, "TOTAL RESOLVED CASES w/NO INTERRUPTIONS = %ld\n", aggregate.rtot_n
on_interrupt);

fprintf(asf, "TOTAL UNRESOLVED CASES w/NO INTERRUPTIONS = %ld\n", aggregate.utot
_non_interrupt);

fprintf(asf, "TOTAL NON-INTERRUPTED CASES = %ld\n", aggregate.tot_non_interrupt);
```

```
    fprintf(asf, "TOTAL RESOLVED CASE w/INTERRUPTIONS = %ld\n", aggregate.rtot_interr
upt);

    fprintf(asf, "TOTAL UNRESOLVED CASE w/INTERRUPTIONS = %ld\n", aggregate.utot_inte
rrupt);

    fprintf(asf, "TOTAL INTERRUPTED CASES = %ld\n", aggregate.tot_interrupt);
    fprintf(asf, "TOTAL SINGLE QUERY CASES = %ld\n", aggregate.tot_cases);
    fprintf(asf, "TOTAL NON-SQ CASES = %ld\n", aggregate.tot_non_cases);
    fprintf(asf, "TOTAL RESOLVED CASES = %ld\n", aggregate.tot_resolved);
    fprintf(asf, "TOTAL UNRESOLVED CASES = %ld\n", aggregate.tot_unresolved);
    fprintf(asf, "TOTAL OUTLIERS = %ld\n", aggregate.tot_outliers);
    fprintf(asf, "TOTAL ADD RECORDS = %ld\n", aggregate.tot_add_records);
    fprintf(asf, "TOTAL RES. ADD MATCHES = %ld\n", aggregate.rtot_add_match);
    fprintf(asf, "TOTAL UNRES. ADD MATCHES = %ld\n", aggregate.utot_add_match);
    fprintf(asf, "TOTAL ADD MATCHES = %ld\n", aggregate.tot_add_matches);

    fprintf(asf, "RUNNING TOTAL OF RES. CASES W/0 ADD MATCHES = %ld\n", aggregate.r
tot_add_match0);

    fprintf(asf, "RUNNING TOTAL OF RES. CASES W/1 ADD MATCHES = %ld\n", aggregate.r
tot_add_match1);

    fprintf(asf, "RUNNING TOTAL OF RES. CASES W/2 ADD MATCHES = %ld\n", aggregate.r
tot_add_match2);

    fprintf(asf, "RUNNING TOTAL OF RES. CASES W/3 ADD MATCHES = %ld\n", aggregate.r
tot_add_match3);

    fprintf(asf, "RUNNING TOTAL OF RES. CASES W/4+ ADD MATCHES = %ld\n", aggregate.
rtot_add_match4_plus);

    fprintf(asf, "RUNNING TOTAL OF UNRES. CASES W/0 ADD MATCHES = %ld\n", aggregate
.utot_add_match0);

    fprintf(asf, "RUNNING TOTAL OF UNRES. CASES W/1 ADD MATCHES = %ld\n", aggregate
.utot_add_match1);

    fprintf(asf, "RUNNING TOTAL OF UNRES. CASES W/2 ADD MATCHES = %ld\n", aggregate
.utot_add_match2);

    fprintf(asf, "RUNNING TOTAL OF UNRES. CASES W/3 ADD MATCHES = %ld\n", aggregate
.utot_add_match3);

    fprintf(asf, "RUNNING TOTAL OF UNRES. CASES W/4+ ADD MATCHES = %ld\n", aggregat
e.utot_add_match4_plus);

    fprintf(asf, "\nAVERAGES AND PERCENTAGES\n\n");

    fprintf(asf, "RUNNING AVG OF BROWSE REPORT TIME = %.2f\n", aggregate.avg_br_tim
e);

    fprintf(asf, "RUNNING AVG OF SQ TIME, RES. W/NO INT = %.2f\n", aggregate.rs_avg
_time);
```



```
fprintf(asf, "RUNNING AVG OF SQ TIME, UNRES. W/NO INT = %.2f\n", aggregate.us_a
vg_time);

fprintf(asf, "RUNNING AVG OF SQ TIME, RES. W/ INT = %.2f\n", aggregate.rsi_avg_
time);

fprintf(asf, "RUNNING AVG OF SQ TIME, UNRES. W/INT = %.2f\n", aggregate.usi_avg
_time);

fprintf(asf, "RUNNING AVG OF SQ BROWSE TIME, RES. W/NO INT = %.2f\n", aggregate
.rq_avg_time);

fprintf(asf, "RUNNING AVG OF SQ BROWSE TIME, UNRES. W/NO INT = %.2f\n", aggrega
te.uq_avg_time);

fprintf(asf, "RUNNING AVG OF SQ BROWSE TIME, RES. W/ INT = %.2f\n", aggregate.r
qi_avg_time);

fprintf(asf, "RUNNING AVG OF SQ BROWSE TIME, UNRES. W/INT = %.2f\n", aggregate.
uqi_tot_time);

fprintf(asf, "RUNNING AVG ADD RECORDS PER BROWSE REP. SELECTION= %.2f\n", aggre
gate.avg_add_record);

fprintf(asf, "RUNNING AVG ADD MATCHES PER RES. CASE= %.2f\n", aggregate.ravg_ad
d_match);

fprintf(asf, "RUNNING AVG ADD MATCHES PER UNRES. CASE= %.2f\n", aggregate.uavg_
add_match);

fprintf(asf, "RUNNING AVG ADD MATCHES PER CASE= %.2f\n", aggregate.avg_add_matc
h);

fprintf(asf, "PERCENTAGE OF RES. CASES W/0 ADD MATCHES = %.2f\n", aggregate.rpe
r_add_match0);

fprintf(asf, "PERCENTAGE OF RES. CASES W/1 ADD MATCHES = %.2f\n", aggregate.rpe
r_add_match1);

fprintf(asf, "PERCENTAGE OF RES. CASES W/2 ADD MATCHES = %.2f\n", aggregate.rpe
r_add_match2);

fprintf(asf, "PERCENTAGE OF RES. CASES W/3 ADD MATCHES = %.2f\n", aggregate.rpe
r_add_match3);

fprintf(asf, "PERCENTAGE OF RES. CASES W/4+ ADD MATCHES = %.2f\n", aggregate.rp
er_add_match4_plus);

fprintf(asf, "PERCENTAGE OF UNRES. CASES W/0 ADD MATCHES = %.2f\n", aggregate.u
per_add_match0);

fprintf(asf, "PERCENTAGE OF UNRES. CASES W/1 ADD MATCHES = %.2f\n", aggregate.u
per_add_match1);

fprintf(asf, "PERCENTAGE OF UNRES. CASES W/2 ADD MATCHES = %.2f\n", aggregate.u
per_add_match2);

fprintf(asf, "PERCENTAGE OF UNRES. CASES W/3 ADD MATCHES = %.2f\n", aggregate.u
per_add_match3);

fprintf(asf, "PERCENTAGE OF UNRES. CASES W/4+ ADD MATCHES = %.2f\n", aggregate.
uper_add_match4_plus);
```

```
fprintf(asf, "\nUI USAGE TOTALS PER SELECTED OPERATION\n\n");
y);
fprintf(asf, "TOTAL SINGLE QUERY SELECTIONS = %ld\n", aggregate.tot_single_query);
fprintf(asf, "TOTAL BROWSE REPORT SELECTIONS = %ld\n", aggregate.tot_browse_report);
fprintf(asf, "TOTAL PRINT REPORT SELECTIONS = %ld\n", aggregate.tot_print_report);
fprintf(asf, "TOTAL BLANKET REPORT SELECTIONS = %ld\n", aggregate.tot_blanket);
fprintf(asf, "TOTAL DIFFERENT REPORT = %ld\n", aggregate.tot_diff_report);
fprintf(asf, "TOTAL QP = %ld\n", aggregate.tot_qp);
fprintf(asf, "TOTAL EMPLOYER DETAIL = %ld\n", aggregate.tot_er_detail);
fprintf(asf, "TOTAL EMPLOYEE DETAIL = %ld\n", aggregate.tot_ee_detail);
fprintf(asf, "TOTAL FINAL TOTALS = %ld\n", aggregate.tot_final);
er_detail);
fprintf(asf, "TOTAL PRINT EMPLOYER DETAIL SELECTIONS= %ld\n", aggregate.tot_pr_ee_detail);
ee_detail);
fprintf(asf, "TOTAL EMPLOYEE DETAILS PRINTED= %ld\n", aggregate.tot_ee_det_printed);
fprintf(asf, "TOTAL PRINT FINAL TOTALS SELECTIONS= %ld\n", aggregate.tot_pr_final);
al);
fprintf(asf, "TOTAL PRINT BLANKET REPORT SELECTIONS = %ld\n", aggregate.tot_pr_blanket);
fprintf(asf, "TOTAL REPORTS PRINTED = %ld\n", aggregate.tot_pr_report);
fclose(asf);

printf("STATS HAVE BEEN WRITTEN TO THE ASCII FILE %s\n\n", ASTATFILE);

    } //end of display and write
    } //end of display and write stats to text file
    } //end of uf != NULL
    } //end of get user number
    } //end of sf != NULL
} //END OF MAIN
```

```
//custom.c
//created 10/12/93
//holds user defined functions used in usercode.c
//Laura L. Downey, NIST, computer scientist
```

```
#include <WINDOWS.H>
#include "GENERIC.H"
#include <fcntl.h>
#include <io.h>
```

```
// STRING FUNCTIONS
```

```
/******CREATEHEADERSTRING******/
void CreateHeaderString()      /* added 6/19/92 by LLD */
                                /* function creates browse header info */
```

```
{
HeaderString[0] = 0;
strcat(HeaderString, "\n RPT-YR: ");
strcat(HeaderString, CurrEmprInfo.ReportYear);
strcat(HeaderString, " RPT-NO: ");
strcat(HeaderString, CurrEmprInfo.seq_no);
strcat(HeaderString, " EIN: ");
strcat(HeaderString, CurrEmprInfo.EIN);
strcat(HeaderString, " TYPE: ");
strcat(HeaderString, CurrEmprInfo.TypeEmpr);
strcat(HeaderString, "\n\n ");
strcat(HeaderString, CurrEmprInfo.EmprName);
strcat(HeaderString, "\n ");
strcat(HeaderString, CurrEmprInfo.EmprStreetAdd);
strcat(HeaderString, "\n ");
strcat(HeaderString, CurrEmprInfo.EmprCity);
strcat(HeaderString, ", ");
strcat(HeaderString, CurrEmprInfo.EmprState);
strcat(HeaderString, " ");
strcat(HeaderString, CurrEmprInfo.EmprZipCode);
} //END OF CREATEHEADERSTRING
```

```
/******CREATEHDETAILSTRING******/
void CreateHDetailString()    /* added 07/01/92 by LLD */
                                /* function creates detail header info */
```

```
{
HDetailString[0] = 0;
strcat(HDetailString, CurrEmprInfo.EmprName);
strcat(HDetailString, "\tRPT-NO: ");
strcat(HDetailString, CurrEmprInfo.seq_no);

strcat(HDetailString, "\n");
strcat(HDetailString, CurrEmprInfo.EmprStreetAdd);
strcat(HDetailString, "\n");
strcat(HDetailString, CurrEmprInfo.EmprCity);
strcat(HDetailString, ", ");
strcat(HDetailString, CurrEmprInfo.EmprState);
strcat(HDetailString, " ");
strcat(HDetailString, CurrEmprInfo.EmprZipCode);
strcat(HDetailString, "\n\n");
strcat(HDetailString, "EIN: ");
strcat(HDetailString, CurrEmprInfo.EIN);
strcat(HDetailString, "\t\tTape Library Number: ");
```



```
strcat (HDetailString, CurrEmprInfo.TapeLibNum);
strcat (HDetailString, "\n");
strcat (HDetailString, "Starting MRN:  ");
strcat (HDetailString, CurrEmprInfo.MRN);
strcat (HDetailString, "\t\tOther EIN:  ");
strcat (HDetailString, CurrEmprInfo.OtherEIN);
strcat (HDetailString, "\n");
strcat (HDetailString, "Ending MRN:  ");
strcat (HDetailString, CurrEmprInfo.EndMRN);
strcat (HDetailString, "\t\tEstab Number:  ");
strcat (HDetailString, CurrEmprInfo.EstabNumber);
strcat (HDetailString, "\n");
strcat (HDetailString, "Report Year:  ");
strcat (HDetailString, CurrEmprInfo.ReportYear);
strcat (HDetailString, "\t\t\tType of Employment:  ");
strcat (HDetailString, CurrEmprInfo.TypeEmpr);
strcat (HDetailString, "\n");
strcat (HDetailString, "Process Year:  ");
strcat (HDetailString, CurrEmprInfo.ProcessYear);
strcat (HDetailString, "\t\tName Code:  ");
strcat (HDetailString, CurrEmprInfo.NameCode);
strcat (HDetailString, "\n");
```

/\*\* this section added 7/16/92 by LLD - produces a string to be sent to the printer \*\*/

```
PHDetailString[0] = 0;
strcpy (PHDetailString, "EMPLOYER HEADER DETAIL\n\n");
strcat (PHDetailString, CurrEmprInfo.EmprName);
strcat (PHDetailString, "\t\tRPT-NO:  ");
strcat (PHDetailString, CurrEmprInfo.seq_no);

strcat (PHDetailString, "\n");
strcat (PHDetailString, CurrEmprInfo.EmprStreetAdd);
strcat (PHDetailString, "\n");
strcat (PHDetailString, CurrEmprInfo.EmprCity);
strcat (PHDetailString, ",  ");
strcat (PHDetailString, CurrEmprInfo.EmprState);
strcat (PHDetailString, "  ");
strcat (PHDetailString, CurrEmprInfo.EmprZipCode);
strcat (PHDetailString, "\n\n");
strcat (PHDetailString, "EIN:  ");
strcat (PHDetailString, CurrEmprInfo.EIN);
strcat (PHDetailString, "\t\t\tTape Library Number:  ");
strcat (PHDetailString, CurrEmprInfo.TapeLibNum);
strcat (PHDetailString, "\n");
strcat (PHDetailString, "Starting MRN:  ");
strcat (PHDetailString, CurrEmprInfo.MRN);
strcat (PHDetailString, "\t\t\tOther EIN:  ");
strcat (PHDetailString, CurrEmprInfo.OtherEIN);
strcat (PHDetailString, "\n");
strcat (PHDetailString, "Ending MRN:  ");
strcat (PHDetailString, CurrEmprInfo.EndMRN);
strcat (PHDetailString, "\t\tEstab Number:  ");
strcat (PHDetailString, CurrEmprInfo.EstabNumber);
strcat (PHDetailString, "\n");
strcat (PHDetailString, "Report Year:  ");
strcat (PHDetailString, CurrEmprInfo.ReportYear);
strcat (PHDetailString, "\t\t\tType of Employment:  ");
strcat (PHDetailString, CurrEmprInfo.TypeEmpr);
strcat (PHDetailString, "\n");
strcat (PHDetailString, "Process Year:  ");
strcat (PHDetailString, CurrEmprInfo.ProcessYear);
strcat (PHDetailString, "\t\t\tName Code:  ");
strcat (PHDetailString, CurrEmprInfo.NameCode);
```

```
} //END OF CREATEHDETAILSTRING
```

```
/******CREATEEDETAIL******/
```

```
void CreateEDetail() /* added 07/06/92 by LLD */  
/* function creates detail employee info */
```

```
{  
EDetailString[0] = 0;  
strcat(EDetailString, "SSN: ");  
strcat(EDetailString, EDetail.EmpSSN);  
strcat(EDetailString, "\t\t\tRPT-NO: ");  
strcat(EDetailString, CurrEmpInfo.seq_no);  
  
strcat(EDetailString, "\n\n");  
strcat(EDetailString, EDetail.EmpName);  
strcat(EDetailString, "\n");  
strcat(EDetailString, EDetail.EmpStreetAdd);  
strcat(EDetailString, "\n");  
strcat(EDetailString, EDetail.EmpCity);  
strcat(EDetailString, ", ");  
strcat(EDetailString, EDetail.EmpState);  
strcat(EDetailString, " ");  
strcat(EDetailString, EDetail.EmpZipCode);  
strcat(EDetailString, "\n\n\n");  
strcat(EDetailString, "FICA Wages: ");  
strcat(EDetailString, EDetail.AnnFICAWages);  
strcat(EDetailString, "\t\tPension Indicator: ");  
strcat(EDetailString, EDetail.PensionInd);  
strcat(EDetailString, "\n");  
strcat(EDetailString, "FICA Tips: ");  
strcat(EDetailString, EDetail.AnnFICATips);  
strcat(EDetailString, "\t\tDefComp Indicator: ");  
strcat(EDetailString, EDetail.DefCompInd);  
strcat(EDetailString, "\n");  
strcat(EDetailString, "Wgs\Tps\Other: ");  
strcat(EDetailString, EDetail.AnnWgsTpsOther);  
strcat(EDetailString, "\t\tDef Comp Amt: ");  
strcat(EDetailString, EDetail.DefCompAmt);  
strcat(EDetailString, "\n");  
strcat(EDetailString, "FICA Tax: ");  
strcat(EDetailString, EDetail.FICATaxWheld);  
strcat(EDetailString, "\t\t\tSta: ");  
strcat(EDetailString, EDetail.StatEmpCode);  
strcat(EDetailString, "\n");  
strcat(EDetailString, "Federal Tax: ");  
strcat(EDetailString, EDetail.FedTaxWheld);  
strcat(EDetailString, "\t\tFr Benefits: ");  
strcat(EDetailString, EDetail.FringeBenefits);  
strcat(EDetailString, "\n");  
strcat(EDetailString, "Uncollected FICA: ");  
strcat(EDetailString, EDetail.UncFICATipTax);  
strcat(EDetailString, "\t\tEmp Grp Trm Ins Cost: ");  
strcat(EDetailString, EDetail.EmpGrpTrmInsCost);  
strcat(EDetailString, "\n");  
strcat(EDetailString, "Medicare Wages: ");  
strcat(EDetailString, EDetail.MedWages);  
strcat(EDetailString, "\tDep Care: ");  
strcat(EDetailString, EDetail.DepCare);  
strcat(EDetailString, "\n");  
strcat(EDetailString, "Medicare Tax: ");  
strcat(EDetailString, EDetail.MedTax);  
strcat(EDetailString, "\t\tNQSEC: ");
```

```
strcat(EDetailString,EDetail.ngsec);
strcat(EDetailString,"\n");
strcat(EDetailString,"Allocated Tips: ");
strcat(EDetailString,EDetail.AllocTips);
strcat(EDetailString,"\t\tNQNOT: ");
strcat(EDetailString,EDetail.ngnot);
strcat(EDetailString,"\n");
strcat(EDetailString,"Adv Earn Inc: ");
strcat(EDetailString,EDetail.AdvEarnInc);
strcat(EDetailString,"\t\tCtrl Number: ");
strcat(EDetailString,EDetail.ControlNumber);
```

/\*\* this section added 7/16/92 by LLD - produces a string to be sent to the printer \*\*/

```
PEDetailString[0] = 0;
strcat(PEDetailString,"SSN: ");
strcat(PEDetailString,EDetail.EmpSSN);
strcat(PEDetailString,"\t\t\tRPT-NO: ");
strcat(PEDetailString,CurrEmprInfo.seq_no);

strcat(PEDetailString,"\n\n");
strcat(PEDetailString,EDetail.EmpName);
strcat(PEDetailString,"\n");
strcat(PEDetailString,EDetail.EmpStreetAdd);
strcat(PEDetailString,"\n");
strcat(PEDetailString,EDetail.EmpCity);
strcat(PEDetailString,", ");
strcat(PEDetailString,EDetail.EmpState);
strcat(PEDetailString," ");
strcat(PEDetailString,EDetail.EmpZipCode);
strcat(PEDetailString,"\n\n\n");
strcat(PEDetailString,"FICA Wages: ");
strcat(PEDetailString,EDetail.AnnFICAWages);
strcat(PEDetailString,"\t\t\tPension Indicator: ");
strcat(PEDetailString,EDetail.PensionInd);
strcat(PEDetailString,"\n");
strcat(PEDetailString,"FICA Tips: ");
strcat(PEDetailString,EDetail.AnnFICATips);
strcat(PEDetailString,"\t\t\tDefComp Indicator: ");
strcat(PEDetailString,EDetail.DefCompInd);
strcat(PEDetailString,"\n");
strcat(PEDetailString,"Wgs\Tps\Other: ");
strcat(PEDetailString,EDetail.AnnWgsTpsOther);
strcat(PEDetailString,"\t\t\tDef Comp Amt: ");
strcat(PEDetailString,EDetail.DefCompAmt);
strcat(PEDetailString,"\n");
strcat(PEDetailString,"FICA Tax: ");
strcat(PEDetailString,EDetail.FICATaxWheld);
strcat(PEDetailString,"\t\t\tSta: ");
strcat(PEDetailString,EDetail.StatEmpCode);
strcat(PEDetailString,"\n");
strcat(PEDetailString,"Federal Tax: ");
strcat(PEDetailString,EDetail.FedTaxWheld);
strcat(PEDetailString,"\t\t\tFr Benefits: ");
strcat(PEDetailString,EDetail.FringeBenefits);
strcat(PEDetailString,"\n");
strcat(PEDetailString,"Uncollected FICA: ");
strcat(PEDetailString,EDetail.UncFICATipTax);
strcat(PEDetailString,"\t\t\tEmpr Grp Trm Ins Cost: ");
strcat(PEDetailString,EDetail.EmprGrpTrmInsCost);
strcat(PEDetailString,"\n");
strcat(PEDetailString,"Medicare Wages: ");
strcat(PEDetailString,EDetail.MedWages);
```



```
    strcat(PEDetailString, "\t\tDep Care:  ");
    strcat(PEDetailString, EDetail.DepCare);
    strcat(PEDetailString, "\n");
    strcat(PEDetailString, "Medicare Tax:  ");
    strcat(PEDetailString, EDetail.MedTax);
    strcat(PEDetailString, "\t\t\tNQSEC:  ");
    strcat(PEDetailString, EDetail.nqsec);
    strcat(PEDetailString, "\n");
    strcat(PEDetailString, "Allocated Tips:  ");
    strcat(PEDetailString, EDetail.AllocTips);
    strcat(PEDetailString, "\t\t\tNQNOT:  ");
    strcat(PEDetailString, EDetail.nqnot);
    strcat(PEDetailString, "\n");
    strcat(PEDetailString, "Adv Earn Inc:  ");
    strcat(PEDetailString, EDetail.AdvEarnInc);
    strcat(PEDetailString, "\t\t\tCtrl Number:  ");
    strcat(PEDetailString, EDetail.ControlNumber);

}    //END OF CREATEEDETAIL

/*****CREATETITLEPAGE*****/
void CreateTitlePage(PTitle)    /** function creates a title page for blanket report */
char PTitle[800];
{
    PTitle[0] = 0;
    strcat(PTitle, "BLANKET REPORT\n\n");
    strcat(PTitle, CurrEmprInfo.EmprName);
    strcat(PTitle, "\n");
    strcat(PTitle, CurrEmprInfo.EmprStreetAdd);
    strcat(PTitle, "\n");
    strcat(PTitle, CurrEmprInfo.EmprCity);
    strcat(PTitle, ",  ");
    strcat(PTitle, CurrEmprInfo.EmprState);
    strcat(PTitle, "  ");
    strcat(PTitle, CurrEmprInfo.EmprZipCode);
    strcat(PTitle, "\n\nRPT-YR: \t");
    strcat(PTitle, CurrEmprInfo.ReportYear);
    strcat(PTitle, "\nRPT-NO: \t");
    strcat(PTitle, CurrEmprInfo.seq_no);

    strcat(PTitle, "\nEST: \t");
    strcat(PTitle, CurrEmprInfo.EstabNumber);
    strcat(PTitle, "\nEIN: \t");
    strcat(PTitle, CurrEmprInfo.EIN);
    strcat(PTitle, "\nTYPE: \t");
    strcat(PTitle, CurrEmprInfo.TypeEmpr);
    CreateTotalString();
    strcat(PTitle, "\n\n");
    strcat(PTitle, PTotalString);
}    //END OF CREATETITLEPAGE

/*****CREATETOTALSTRING*****/
void CreateTotalString()    /** added 8/19/92 by LLD */
    /** create a total string for display and printing */
{
    TotalString[0] = 0;
```

```
strcat(TotalString, " FICA Wages\t\t");
strcat(TotalString, CurrEmprInfo.ProcFICAWages);
strcat(TotalString, "\t\t");
strcat(TotalString, CurrEmprInfo.RepFICAWages);
strcat(TotalString, "\n\n ");
strcat(TotalString, "FICA Tips\t\t");
strcat(TotalString, CurrEmprInfo.ProcFICATips);
strcat(TotalString, "\t\t");
strcat(TotalString, CurrEmprInfo.RepFICATips);
strcat(TotalString, "\n\n ");
strcat(TotalString, "Wgs/Tps/Other\t\t");
strcat(TotalString, CurrEmprInfo.ProcWgsTpsOther);
strcat(TotalString, "\t\t");
strcat(TotalString, CurrEmprInfo.RepWgsTpsOther);
strcat(TotalString, "\n\n ");
strcat(TotalString, "FICA Tax W/H\t\t");
strcat(TotalString, CurrEmprInfo.ProcFICATaxWheld);
strcat(TotalString, "\t\t");
strcat(TotalString, CurrEmprInfo.RepFICATaxWheld);
strcat(TotalString, "\n\n ");
strcat(TotalString, "Medicare Wgs\t\t");
strcat(TotalString, CurrEmprInfo.ProcMedWages);
strcat(TotalString, "\t\t");
strcat(TotalString, CurrEmprInfo.RepMedWages);
strcat(TotalString, "\n\n ");
strcat(TotalString, "Medicare Tax\t\t");
strcat(TotalString, CurrEmprInfo.ProcMedTax);
strcat(TotalString, "\t\t");
strcat(TotalString, CurrEmprInfo.RepMedTax);
strcat(TotalString, "\n\n ");
strcat(TotalString, "Number of Items\t\t");
strcat(TotalString, CurrEmprInfo.ProcItems);
strcat(TotalString, "\t\t\t");
strcat(TotalString, CurrEmprInfo.RepItems);
strcat(TotalString, "\n\n\n ");
strcat(TotalString, "Fed Tax W/H\t\t");
strcat(TotalString, CurrEmprInfo.ProcFedTaxWheld);
strcat(TotalString, "\t\t");
strcat(TotalString, CurrEmprInfo.RepFedTaxWheld);
strcat(TotalString, "\n\n ");
strcat(TotalString, "Earned Income\t\t");
strcat(TotalString, CurrEmprInfo.ProcEarnInc);
strcat(TotalString, "\t\t");
strcat(TotalString, CurrEmprInfo.RepEarnInc);
strcat(TotalString, "\n\n ");
strcat(TotalString, "DefComp\t\t");
strcat(TotalString, CurrEmprInfo.ProcDefComp);
strcat(TotalString, "\t\t");
strcat(TotalString, CurrEmprInfo.RepDefComp);
strcat(TotalString, "\n\n ");
strcat(TotalString, "NonEqual\t\t");
strcat(TotalString, CurrEmprInfo.ProcNonequal);
strcat(TotalString, "\t\t");
strcat(TotalString, CurrEmprInfo.RepNonequal);

PTotalString[0] = 0;

strcat(PTotalString, "TYPE:\t\t\tPROCESSED:\t\t\tREPORTED:\n\n");
strcat(PTotalString, "FICA Wages\t\t");
strcat(PTotalString, CurrEmprInfo.ProcFICAWages);
strcat(PTotalString, "\t\t");
strcat(PTotalString, CurrEmprInfo.RepFICAWages);
strcat(PTotalString, "\n");
```

```
strcat (PTotalString, "FICA Tips\t\t");
strcat (PTotalString, CurrEmprInfo.ProcFICATips);
strcat (PTotalString, "\t\t");
strcat (PTotalString, CurrEmprInfo.RepFICATips);
strcat (PTotalString, "\n");
strcat (PTotalString, "Wgs/Tps/Other\t\t");
strcat (PTotalString, CurrEmprInfo.ProcWgsTpsOther);
strcat (PTotalString, "\t\t");
strcat (PTotalString, CurrEmprInfo.RepWgsTpsOther);
strcat (PTotalString, "\n");
strcat (PTotalString, "FICA Tax W/H\t\t");
strcat (PTotalString, CurrEmprInfo.ProcFICATaxWheld);
strcat (PTotalString, "\t\t");
strcat (PTotalString, CurrEmprInfo.RepFICATaxWheld);
strcat (PTotalString, "\n");
strcat (PTotalString, "Medicare Wgs\t\t");
strcat (PTotalString, CurrEmprInfo.ProcMedWages);
strcat (PTotalString, "\t\t");
strcat (PTotalString, CurrEmprInfo.RepMedWages);
strcat (PTotalString, "\n");
strcat (PTotalString, "Medicare Tax\t\t");
strcat (PTotalString, CurrEmprInfo.ProcMedTax);
strcat (PTotalString, "\t\t");
strcat (PTotalString, CurrEmprInfo.RepMedTax);
strcat (PTotalString, "\n");
strcat (PTotalString, "Number of Items\t\t");
strcat (PTotalString, CurrEmprInfo.ProcItems);
strcat (PTotalString, "\t\t\t");
strcat (PTotalString, CurrEmprInfo.RepItems);
strcat (PTotalString, "\n\n");
strcat (PTotalString, "Fed Tax W/H\t\t");
strcat (PTotalString, CurrEmprInfo.ProcFedTaxWheld);
strcat (PTotalString, "\t\t");
strcat (PTotalString, CurrEmprInfo.RepFedTaxWheld);
strcat (PTotalString, "\n");
strcat (PTotalString, "Earned Income\t\t");
strcat (PTotalString, CurrEmprInfo.ProcEarnInc);
strcat (PTotalString, "\t\t");
strcat (PTotalString, CurrEmprInfo.RepEarnInc);
strcat (PTotalString, "\n");
strcat (PTotalString, "DefComp\t\t\t");
strcat (PTotalString, CurrEmprInfo.ProcDefComp);
strcat (PTotalString, "\t\t");
strcat (PTotalString, CurrEmprInfo.RepDefComp);
strcat (PTotalString, "\n");
strcat (PTotalString, "NonEqual\t\t");
strcat (PTotalString, CurrEmprInfo.ProcNonequal);
strcat (PTotalString, "\t\t");
strcat (PTotalString, CurrEmprInfo.RepNonequal);

} //END OF CREATETOTALSTRING
```

```
/******CREATEPRINTTOTSTRING*****
/
void CreatePrintTotString()          /** added 8/19/92 by LLD **/
                                     /** create a total string for printing **/
{
    char Title[] = "REPORT TOTALS\n\n"; /** title line for report total printout **/

    PrintTotString[0] = 0;
    strcat (PrintTotString, Title);   /** concatenate title **/
}
```



```
CreateHDetailString();          /** create header info **/
strcat(PrintTotString,PHDetailString); /** concatenate header info **/
strcat(PrintTotString,"\n");
strcat(PrintTotString,separatorstring); /** concatenate separator string **/
strcat(PrintTotString,"\n");
strcat(PrintTotString,PTotalString); /** concatenate total print string **/
}      //END OF CREATEPRINTTOTSTRING
```

```
/******CREATEPRINTDETAIL******/
void CreatePrintEDetail()      /** added 8/19/92 by LLD **/
                                /** creates an employee detail for printing **/
```

```
{
char Title[] = "EMPLOYEE DETAIL\n\n";          /** title line for report total printout *
*/
```

```
PrintEDetail[0] = 0;
strcat(PrintEDetail,Title);
CreateHDetailString();
strcat(PrintEDetail,PHDetailString);
strcat(PrintEDetail,"\n");
strcat(PrintEDetail,separatorstring);
strcat(PrintEDetail,"\n");
strcat(PrintEDetail,PEDetailString);
```

```
}      //END OF CREATEPRINTDETAIL
```

```
/******COPYBLANKET_EDetail******/
void CopyBlanket_EDetail(index) /** copies current blanket detail to EDetail **/
```

```
int index;          /** passed parameter **/
```

```
{
strcpy(EDetail.MRN, Blanket[index].MRN);
strcpy(EDetail.EmpSSN, Blanket[index].EmpSSN);
strcpy(EDetail.EmpName, Blanket[index].EmpName);
strcpy(EDetail.PensionInd, Blanket[index].PensionInd);
strcpy(EDetail.DefCompInd, Blanket[index].DefCompInd);
strcpy(EDetail.AnnFICAWages, Blanket[index].AnnFICAWages);
strcpy(EDetail.AnnFICATips, Blanket[index].AnnFICATips);
strcpy(EDetail.AnnWgsTpsOther, Blanket[index].AnnWgsTpsOther);
strcpy(EDetail.FedTaxWheld, Blanket[index].FedTaxWheld);
strcpy(EDetail.FICATaxWheld, Blanket[index].FICATaxWheld);
strcpy(EDetail.AdvEarnInc, Blanket[index].AdvEarnInc);
strcpy(EDetail.MedWages, Blanket[index].MedWages);
strcpy(EDetail.MedTax, Blanket[index].MedTax);
strcpy(EDetail.ControlNumber, Blanket[index].ControlNumber);

strcpy(EDetail.EmpStreetAdd, Blanket[index].EmpStreetAdd);
strcpy(EDetail.DepCare, Blanket[index].DepCare);
strcpy(EDetail.AllocTips, Blanket[index].AllocTips);
strcpy(EDetail.EmprGrpTrmInsCost, Blanket[index].EmprGrpTrmInsCost);
strcpy(EDetail.UncFICATipTax, Blanket[index].UncFICATipTax);

strcpy(EDetail.EmpCity, Blanket[index].EmpCity);
strcpy(EDetail.EmpState, Blanket[index].EmpState);
strcpy(EDetail.EmpZipCode, Blanket[index].EmpZipCode);
strcpy(EDetail.DefCompAmt, Blanket[index].DefCompAmt);
strcpy(EDetail.StatEmpCode, Blanket[index].StatEmpCode);
strcpy(EDetail.FringeBenefits, Blanket[index].FringeBenefits);
strcpy(EDetail.nqsec, Blanket[index].nqsec);
```

```
strcpy(EDetail.nqnot, Blanket[index].nqnot);  
}  
//END OF COPYBLANKET_EDTAIL
```

```
//Filename: EAMAT42.C
// "EAMAT42" Generated by WindowsMAKER Professional
//Author: Laura L. Downey

//
// Code in this file is initially generated by WindowsMAKER Professional.
// This file contains the WINMAIN and the MAINWINPROC functions.
// You can override the functionality supplied by WindowsMAKER Professional
// by adding your own code or replacing calls in this file. For example if
// you want to change the normal flow of events and bring up a login box
// before the main window is displayed, you would add the code here.
// For more information see the section "How Code is Generated" in the
// documentation.

#include <WINDOWS.H>
#include "GENERIC.H"

WMPDEBUG
#include "EAMAT42.WMC"
#include "main.h" //defines variables used for displaying credit window - LLD 6/18/93

extern unsigned_stklen = 43210;
/*****
//
// WinMain FUNCTION
//
int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
    MSG msg; // message

    hInst = hInstance; // Saves the current instance

    if (!BLDInitApplication(hInstance,hPrevInstance,&nCmdShow,lpCmdLine))
        return FALSE;

    if (!hPrevInstance) // Is there an other instance of the task
    {
        if (!BLDRegisterClass(hInstance))
            return FALSE; // Exits if unable to initialize
    }

    MainhWnd = BLDCreateWindow(hInstance);
    if (!MainhWnd) // Check if the window is created
        return FALSE;

    ShowWindow(MainhWnd, nCmdShow); // Show the window
    UpdateWindow(MainhWnd); // Send WM_PAINT message to window

    /***** ADDED SECTION (6/18/93) to display credits in main window *****/
    /***** put About Dialog Box in Main Window *****/
    lpProc = MakeProcInstance((FARPROC)BLD_FunctionDlgProc, hInst);
    AbouthWnd = DialogBox(hInst, (LPSTR)"MAIN1", MainhWnd, lpProc);
    FreeProcInstance(lpProc);
    ShowWindow(AbouthWnd, SW_SHOW);
    UpdateWindow(AbouthWnd);
    /*****

    BLDInitMainMenu(MainhWnd); // Initialize main menu if necessary
```



```
while (GetMessage(&msg,      // message structure
               0,          // handle of window receiving the message
               0,          // lowest message to examine
               0))         // highest message to examine
{
    if (BLDKeyTranslation(&msg)) // WindowsMAKER code for key translation
        continue;
    TranslateMessage(&msg); // Translates character keys
    DispatchMessage(&msg); // Dispatches message to window
}
BLDExitApplication(); // Clean up if necessary
return(msg.wParam);    // Returns the value from PostQuitMessage
}

//*****
//          WINDOW PROCEDURE FOR MAIN WINDOW
//*****

LONG FAR PASCAL BLDMainWndProc(HWND hWnd, UINT message, UINT wParam, LONG lParam)
{
    switch (message)
    {
        case WM_CREATE: // window creation
            // Send to BLDDefWindowProc in (.WMC) for controls in main window
            return BLDDefWindowProc(hWnd, message, wParam, lParam);
            break;

        case WM_SETFOCUS: // window is notified of focus change
            // Send to BLDDefWindowProc in (.WMC) for controls in main window
            return BLDDefWindowProc(hWnd, message, wParam, lParam);
            break;

        case WM_DESTROY: // window being destroyed
            PostQuitMessage(0);
            return BLDDefWindowProc(hWnd, message, wParam, lParam);
            break;

        case WM_COMMAND: // command from the main window
            if (BLDMenuCommand(hWnd, message, wParam, lParam))
                break; // Processed by BLDMenuCommand.
            // else default processing by BLDDefWindowProc.
        default:
            // Pass on message for default processing
            return BLDDefWindowProc(hWnd, message, wParam, lParam);
    }
    return FALSE; // Returns FALSE if processed
}

// *****
//          INIT & EXIT FOR MAIN WINDOW
// *****

BOOL BLDMainRegClass(HINSTANCE hInstance)
{
    return BLDMainRegClassDef(hInstance);
}
```

```
BOOL BLDMainExitClass()  
{  
    return BLDMainExitClassDef();  
}
```

```
HWND BLDMainCreateWnd()  
{  
    return BLDMainCreateWndDef();  
}
```

```
// Filename: ERROR.C
// "EAMAT42" Generated by WindowsMAKER Professional.
// Author: Laura L. Downey, NIST, computer scientist
// created 10/28/93
// holds error dialog and message boxes used in usercode.c
```

```
#include "WINDOWS.H"
#include "GENERIC.H"
```

```
WMPDEBUG
#include "ERROR.WMC"
```

```
// ERROR DIALOGS AND MESSAGE BOXES
```

```
/* error box when employer report can't be found using current EIN */
```

```
// *****
// Modal Dialog Box: EINERROR
// *****
```

```
// Startup procedure for modal dialog box
int BLD_EINErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_EINErrDlgFuncDef(hWnd,(char *)NULL);
}
```

```
// Modal dialog box procedure
BOOL CALLBACK BLD_EINErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_EINErrDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_EINErrDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;

                case IDCANCEL:
                    if (!BLD_EINErrDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;

                default:
                    return BLD_EINErrDlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;
    }
}
```



```
default:
    return BLD_EINErrDlgDefault (hDlg,message,wParam,lParam);
    break;
}
return TRUE;// Did process the message
} //END OF EIN ERROR MESSAGE BOX

/*****
/* displays error message box when user does not enter proper query info */
*****/

// ****
//          Modal Dialog Box: QMSG
// ****

// Startup procedure for modal dialog box
int BLD_QueryMessageDlgFunc (HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_QueryMessageDlgFuncDef (hWnd, (char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_QueryMessageDlgProc (HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch (message)
    {
        case WM_INITDIALOG:
            return BLD_QueryMessageDlgDefault (hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD (wParam);
            switch (wId)
            {
                case IDOK:
                    if (!BLD_QueryMessageDlgDefault (hDlg,message,wParam,lParam))
                        EndDialog (hDlg, IDOK);
                    break;

                case IDCANCEL:
                    if (!BLD_QueryMessageDlgDefault (hDlg,message,wParam,lParam))
                        EndDialog (hDlg, IDCANCEL);
                    break;

                default:
                    return BLD_QueryMessageDlgDefault (hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
            return BLD_QueryMessageDlgDefault (hDlg,message,wParam,lParam);
            break;
    }
    return TRUE;// Did process the message
} //END OF QUERY ERROR MESSAGE BOX
```

```

/*****
    Displays error message if both the first and last name are not
    entered if user wants to search by employee name
*****/

// ****
//      Modal Dialog Box: NMSG
// ****

// Startup procedure for modal dialog box
int BLD_NMSGDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_NMSGDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_NMSGDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
    case WM_INITDIALOG:
        return BLD_NMSGDlgDefault(hDlg,message,wParam,lParam);
        break;

    case WM_COMMAND:
        wId=LOWORD(wParam);
        switch(wId)
        {
            case IDOK:
                if (!BLD_NMSGDlgDefault(hDlg,message,wParam,lParam))
                    EndDialog(hDlg,IDOK);
                break;

            case IDCANCEL:
                if (!BLD_NMSGDlgDefault(hDlg,message,wParam,lParam))
                    EndDialog(hDlg,IDCANCEL);
                break;

            default:
                return BLD_NMSGDlgDefault(hDlg,message,wParam,lParam);
                break;
        }
        break;

    default:
        return BLD_NMSGDlgDefault(hDlg,message,wParam,lParam);
        break;
    }

    return TRUE; // Did process the message
} //END OF BOTH FN & LN ERROR MESSAGE BOX

/*****
** error box that appears when a search error has occurred
** during a single-query search
*****/
```

```
// *****
//           Modal Dialog Box: QUERYERROR
// *****

// Startup procedure for modal dialog box
int BLD_QueryErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_QueryErrDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_QueryErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_QueryErrDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_QueryErrDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;

                case IDCANCEL:
                    if (!BLD_QueryErrDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;

                default:
                    return BLD_QueryErrDlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
            return BLD_QueryErrDlgDefault(hDlg,message,wParam,lParam);
            break;
    }

    return TRUE;// Did process the message
} //END OF SINGLE QUERY ERROR MESSAGE BOX

/*****
** error box that appears when a search error has occurred
concerning the blanket report
**/
*****/

// *****
//           Modal Dialog Box: BLANKETERROR
// *****

// Startup procedure for modal dialog box
int BLD_BlanketErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{

```



```
    return BLD_BlanketErrDlgFuncDef(hWnd, (char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_BlanketErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_BlanketErrDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_BlanketErrDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;

                case IDCANCEL:
                    if (!BLD_BlanketErrDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;

                default:
                    return BLD_BlanketErrDlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
            return BLD_BlanketErrDlgDefault(hDlg,message,wParam,lParam);
            break;
    }
    return TRUE;// Did process the message
} //END OF BLANKET SYSTEM ERROR MESSAGE BOX


/*****
** error box that appears when an incorrect password is entered
*****/

// *****
//                               Modal Dialog Box: PWERROR
// *****

// Startup procedure for modal dialog box
int BLD_PWErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_PWErrDlgFuncDef(hWnd, (char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_PWErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;
```

```
switch(message)
{
case WM_INITDIALOG:
    return BLD_PWErrDlgDefault(hDlg,message,wParam,lParam);
    break;

case WM_COMMAND:
    wId=LOWORD(wParam);
    switch(wId)
    {
    case IDOK:
        if (!BLD_PWErrDlgDefault(hDlg,message,wParam,lParam))
            EndDialog(hDlg,IDOK);
        break;

    case IDCANCEL:
        if (!BLD_PWErrDlgDefault(hDlg,message,wParam,lParam))
            EndDialog(hDlg,IDCANCEL);
        break;

    default:
        return BLD_PWErrDlgDefault(hDlg,message,wParam,lParam);
        break;
    }
    break;

default:
    return BLD_PWErrDlgDefault(hDlg,message,wParam,lParam);
    break;
}

return TRUE; // Did process the message
} //END OF PASSWORD ERROR MESSAGE BOX


/*****
** error message that appears when incorrect year is entered
*****/

// *****
// Modal Dialog Box: YEAR_ERR
// *****

// Startup procedure for modal dialog box
int BLD_Year_ErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_Year_ErrDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_Year_ErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD wId;

    switch(message)
    {
    case WM_INITDIALOG:
        return BLD_Year_ErrDlgDefault(hDlg,message,wParam,lParam);
        break;

    case WM_COMMAND:
        wId=LOWORD(wParam);
        switch(wId)
```

```
{
case IDOK:
    if (!BLD_Year_ErrDlgDefault(hDlg,message,wParam,lParam))
        EndDialog(hDlg,IDOK);
    break;

case IDCANCEL:
    if (!BLD_Year_ErrDlgDefault(hDlg,message,wParam,lParam))
        EndDialog(hDlg,IDCANCEL);
    break;

default:
    return BLD_Year_ErrDlgDefault(hDlg,message,wParam,lParam);
    break;
}
break;

default:
    return BLD_Year_ErrDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE; // Did process the message
} //END OF YEAR ERROR MESSAGE BOX

/*****
** system error message box
**
*****/

// *****
//          Modal Dialog Box: SYSERR
// *****

// Startup procedure for modal dialog box
int BLD_SysErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_SysErrDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_SysErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
    case WM_INITDIALOG:
        return BLD_SysErrDlgDefault(hDlg,message,wParam,lParam);
        break;

    case WM_COMMAND:
        wId=LOWORD(wParam);
        switch(wId)
        {
        case IDOK:
            if (!BLD_SysErrDlgDefault(hDlg,message,wParam,lParam))
                EndDialog(hDlg,IDOK);
            break;

        case IDCANCEL:
            if (!BLD_SysErrDlgDefault(hDlg,message,wParam,lParam))
                EndDialog(hDlg,IDCANCEL);
            break;
        }
    }
}
```



```
        break;

    default:
        return BLD_SysErrDlgDefault(hDlg,message,wParam,lParam);
        break;
    }
    break;

default:
    return BLD_SysErrDlgDefault(hDlg,message,wParam,lParam);
    break;
}

return TRUE; // Did process the message
} //END OF SYSTEM ERROR MESSAGE BOX


/*****
** error message box that appears when blanket files can't be found **
*****/

// *****
//                               Modal Dialog Box: BLANKERR
// *****

// Startup procedure for modal dialog box
int BLD_BlankErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_BlankErrDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_BlankErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_BlankErrDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_BlankErrDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;

                case IDCANCEL:
                    if (!BLD_BlankErrDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;

                default:
                    return BLD_BlankErrDlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;
    }
}
```

```
default:
    return BLD_BlankErrDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE;// Did process the message
} //END OF BLANKET FILES NOT FOUND MESSAGE BOX

/*****
** error message that appears when files can't be found
**
*****/

// *****
// Modal Dialog Box: MISSINGFILE
// *****

// Startup procedure for modal dialog box
int BLD_MissingFileDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_MissingFileDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_MissingFileDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_MissingFileDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_MissingFileDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;

                case IDCANCEL:
                    if (!BLD_MissingFileDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;

                default:
                    return BLD_MissingFileDlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
            return BLD_MissingFileDlgDefault(hDlg,message,wParam,lParam);
            break;
    }
    return TRUE;// Did process the message
} //END OF MISSING FILE MESSAGE BOX
```

```
// appears when query*.txt file can not be opened
// *****
//           Modal Dialog Box: QUERYTXT
// *****

// Startup procedure for modal dialog box
int BLD_QueryTxtDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_QueryTxtDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_QueryTxtDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_QueryTxtDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_QueryTxtDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;

                case IDCANCEL:
                    if (!BLD_QueryTxtDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;

                default:
                    return BLD_QueryTxtDlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
            return BLD_QueryTxtDlgDefault(hDlg,message,wParam,lParam);
            break;
    }

    return TRUE;// Did process the message
} //End of query*.txt file can not be open message

// appears when local error file can't be found
// *****
//           Modal Dialog Box: ERRORFILE
// *****

// Startup procedure for modal dialog box
int BLD_ErrorFileDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_ErrorFileDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
```



```
BOOL CALLBACK BLD_ErrorFileDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_ErrorFileDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_ErrorFileDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;

                case IDCANCEL:
                    if (!BLD_ErrorFileDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;

                default:
                    return BLD_ErrorFileDlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
            return BLD_ErrorFileDlgDefault(hDlg,message,wParam,lParam);
            break;
    }
    return TRUE;// Did process the message
} //end of can't find local error file


// appears when null ptrs occur during WM_INITDIALOG
// *****
//           Modal Dialog Box: NULLPTR
// *****


// Startup procedure for modal dialog box
int BLD_NULLPtrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_NULLPtrDlgFuncDef(hWnd,(char *)NULL);
}


// Modal dialog box procedure
BOOL CALLBACK BLD_NULLPtrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_NULLPtrDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
```

```
{
case IDOK:
    if (!BLD_NULLPtrDlgDefault(hDlg,message,wParam,lParam))
        EndDialog(hDlg,IDOK);
    break;

case IDCANCEL:
    if (!BLD_NULLPtrDlgDefault(hDlg,message,wParam,lParam))
        EndDialog(hDlg,IDCANCEL);
    break;

default:
    return BLD_NULLPtrDlgDefault(hDlg,message,wParam,lParam);
    break;
}
break;

default:
    return BLD_NULLPtrDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE; // Did process the message
} //end of nullptr

// appears when data files can't be opened to add records to the list box
// *****
//                               Modal Dialog Box: DATAERR
// *****

// Startup procedure for modal dialog box
int BLD_DataErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_DataErrDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_DataErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
    case WM_INITDIALOG:
        return BLD_DataErrDlgDefault(hDlg,message,wParam,lParam);
        break;

    case WM_COMMAND:
        wId=LOWORD(wParam);
        switch(wId)
        {
        case IDOK:
            if (!BLD_DataErrDlgDefault(hDlg,message,wParam,lParam))
                EndDialog(hDlg,IDOK);
            break;

        case IDCANCEL:
            if (!BLD_DataErrDlgDefault(hDlg,message,wParam,lParam))
                EndDialog(hDlg,IDCANCEL);
            break;
        }
    }
}
```

```
        default:
            return BLD_DataErrDlgDefault(hDlg,message,wParam,lParam);
            break;
        }
    break;

default:
    return BLD_DataErrDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE;// Did process the message
} //end of dataerr

// appears when detail*.txt can't be opened
// *****
//           Modal Dialog Box: DETAILTXT
// *****

// Startup procedure for modal dialog box
int BLD_DFileErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_DFileErrDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_DFileErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
    case WM_INITDIALOG:
        return BLD_DFileErrDlgDefault(hDlg,message,wParam,lParam);
        break;

    case WM_COMMAND:
        wId=LOWORD(wParam);
        switch(wId)
        {
        case IDOK:
            if (!BLD_DFileErrDlgDefault(hDlg,message,wParam,lParam))
                EndDialog(hDlg,IDOK);
            break;

        case IDCANCEL:
            if (!BLD_DFileErrDlgDefault(hDlg,message,wParam,lParam))
                EndDialog(hDlg,IDCANCEL);
            break;

        default:
            return BLD_DFileErrDlgDefault(hDlg,message,wParam,lParam);
            break;
        }
    break;

default:
    return BLD_DFileErrDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE;// Did process the message
} //end of can't open detail*.txt
```



```
// appears when result of add_matches is -1 indicating no more matches found
// *****
//           Modal Dialog Box: NOMORE
// *****

// Startup procedure for modal dialog box
int BLD_NoMoreMatchesDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_NoMoreMatchesDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_NoMoreMatchesDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_NoMoreMatchesDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_NoMoreMatchesDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;

                case IDCANCEL:
                    if (!BLD_NoMoreMatchesDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;

                default:
                    return BLD_NoMoreMatchesDlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
            return BLD_NoMoreMatchesDlgDefault(hDlg,message,wParam,lParam);
            break;
    }

    return TRUE;// Did process the message
} //end of no more matches

// appears when incorrect MRN is entered during browse report
// *****
//           Modal Dialog Box: MRNERR
// *****

// Startup procedure for modal dialog box
int BLD_MRNErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_MRNErrDlgFuncDef(hWnd,(char *)NULL);
}
```

```
// Modal dialog box procedure
BOOL CALLBACK BLD_MRNErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
    case WM_INITDIALOG:
        return BLD_MRNErrDlgDefault(hDlg,message,wParam,lParam);
        break;

    case WM_COMMAND:
        wId=LOWORD(wParam);
        switch(wId)
        {
        case IDOK:
            if (!BLD_MRNErrDlgDefault(hDlg,message,wParam,lParam))
                EndDialog(hDlg,IDOK);
            break;

        case IDCANCEL:
            if (!BLD_MRNErrDlgDefault(hDlg,message,wParam,lParam))
                EndDialog(hDlg,IDCANCEL);
            break;

        default:
            return BLD_MRNErrDlgDefault(hDlg,message,wParam,lParam);
            break;
        }
        break;

    default:
        return BLD_MRNErrDlgDefault(hDlg,message,wParam,lParam);
        break;
    }
    return TRUE;// Did process the message
} //end of error box for incorrect MRN


// appears when no matches are returned during single query
// *****
// Modal Dialog Box: NOMATCH
// *****


// Startup procedure for modal dialog box
int BLD_MatchErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_MatchErrDlgFuncDef(hWnd,(char *)NULL);
}


// Modal dialog box procedure
BOOL CALLBACK BLD_MatchErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
    case WM_INITDIALOG:
        return BLD_MatchErrDlgDefault(hDlg,message,wParam,lParam);
        break;
```

```
case WM_COMMAND:
    wId=LOWORD(wParam);
    switch(wId)
    {
        case IDOK:
            if (!BLD_MatchErrDlgDefault(hDlg,message,wParam,lParam))
                EndDialog(hDlg,IDOK);
            break;

        case IDCANCEL:
            if (!BLD_MatchErrDlgDefault(hDlg,message,wParam,lParam))
                EndDialog(hDlg,IDCANCEL);
            break;

        default:
            return BLD_MatchErrDlgDefault(hDlg,message,wParam,lParam);
            break;
    }
    break;

default:
    return BLD_MatchErrDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE; // Did process the message
} //end of error box with message of no matches to this query


// appears when EIN or sequence number is incorrect during print report
// *****
//                      Modal Dialog Box: EINORSEQ
// *****

// Startup procedure for modal dialog box
int BLD_EINorSeqErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_EINorSeqErrDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_EINorSeqErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_EINorSeqErrDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_EINorSeqErrDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;
            }
    }
}
```



```
case IDCANCEL:
    if (!BLD_EINorSeqErrDlgDefault(hDlg,message,wParam,lParam))
        EndDialog(hDlg,IDCANCEL);
    break;

default:
    return BLD_EINorSeqErrDlgDefault(hDlg,message,wParam,lParam);
    break;
}
break;

default:
    return BLD_EINorSeqErrDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE; // Did process the message
} //end of incorrect EIN or sequence number during print report

// appears when sequence number is incorrect during change header of potential blanket
// *****
//                               Modal Dialog Box: SEQERR
// *****

// Startup procedure for modal dialog box
int BLD_SeqErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_SeqErrDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_SeqErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
    case WM_INITDIALOG:
        return BLD_SeqErrDlgDefault(hDlg,message,wParam,lParam);
        break;

    case WM_COMMAND:
        wId=LOWORD(wParam);
        switch(wId)
        {
            case IDOK:
                if (!BLD_SeqErrDlgDefault(hDlg,message,wParam,lParam))
                    EndDialog(hDlg,IDOK);
                break;

            case IDCANCEL:
                if (!BLD_SeqErrDlgDefault(hDlg,message,wParam,lParam))
                    EndDialog(hDlg,IDCANCEL);
                break;

            default:
                return BLD_SeqErrDlgDefault(hDlg,message,wParam,lParam);
                break;
        }
    }
    break;
}
```

```
default:
    return BLD_SeqErrDlgDefault (hDlg,message,wParam,lParam);
    break;
}
return TRUE;// Did process the message
} //end of incorrect sequence number during change header or potential blanket

// appears when header*.txt can't be opened during change header
// *****
// Modal Dialog Box: HEADERTXT
// *****

// Startup procedure for modal dialog box
int BLD_HFileDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_HFileDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_HFileDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_HFileDlgDefault (hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_HFileDlgDefault (hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;

                case IDCANCEL:
                    if (!BLD_HFileDlgDefault (hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;

                default:
                    return BLD_HFileDlgDefault (hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
            return BLD_HFileDlgDefault (hDlg,message,wParam,lParam);
            break;
    }
    return TRUE;// Did process the message
} //end of header*.txt can't be opened during change header
```

```
// appears when usernum.fil can't be accessed
// *****
//           Modal Dialog Box: USERERR
// *****

// Startup procedure for modal dialog box
int BLD_UserNumErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_UserNumErrDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_UserNumErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_UserNumErrDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_UserNumErrDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;

                case IDCANCEL:
                    if (!BLD_UserNumErrDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;

                default:
                    return BLD_UserNumErrDlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
            return BLD_UserNumErrDlgDefault(hDlg,message,wParam,lParam);
            break;
    }

    return TRUE;// Did process the message
} //end of no access to usernum.fil

// *****
//           Modal Dialog Box: STATOPEN
// *****

// Startup procedure for modal dialog box
int BLD_StatOpenDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_StatOpenDlgFuncDef(hWnd,(char *)NULL);
}
```



```
// Modal dialog box procedure
BOOL CALLBACK BLD_StatOpenDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_StatOpenDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_StatOpenDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;
                case IDCANCEL:
                    if (!BLD_StatOpenDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;
                default:
                    return BLD_StatOpenDlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
            return BLD_StatOpenDlgDefault(hDlg,message,wParam,lParam);
            break;
    }
    return TRUE; // Did process the message
}

// *****
//                      Modal Dialog Box: STATEMPTY
// *****

// Startup procedure for modal dialog box
int BLD_StatEmptyDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_StatEmptyDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_StatEmptyDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_StatEmptyDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
```

```
        if (!BLD_StatEmptyDlgDefault(hDlg,message,wParam,lParam))
            EndDialog(hDlg,IDOK);
        break;
    case IDCANCEL:
        if (!BLD_StatEmptyDlgDefault(hDlg,message,wParam,lParam))
            EndDialog(hDlg,IDCANCEL);
        break;
    default:
        return BLD_StatEmptyDlgDefault(hDlg,message,wParam,lParam);
        break;
    }
    break;

default:
    return BLD_StatEmptyDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE; // Did process the message
}

// *****
//           Modal Dialog Box: STATWOPEN
// *****

// Startup procedure for modal dialog box
int BLD_StatWOpenDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_StatWOpenDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_StatWOpenDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
    case WM_INITDIALOG:
        return BLD_StatWOpenDlgDefault(hDlg,message,wParam,lParam);
        break;

    case WM_COMMAND:
        wId=LOWORD(wParam);
        switch(wId)
        {
        case IDOK:
            if (!BLD_StatWOpenDlgDefault(hDlg,message,wParam,lParam))
                EndDialog(hDlg,IDOK);
            break;
        case IDCANCEL:
            if (!BLD_StatWOpenDlgDefault(hDlg,message,wParam,lParam))
                EndDialog(hDlg,IDCANCEL);
            break;
        default:
            return BLD_StatWOpenDlgDefault(hDlg,message,wParam,lParam);
            break;
        }
        break;

    default:
        return BLD_StatWOpenDlgDefault(hDlg,message,wParam,lParam);
        break;
    }
}
```

```
    }
    return TRUE; // Did process the message
}

// *****
//           Modal Dialog Box: STATWRITE
// *****

// Startup procedure for modal dialog box
int BLD_StatWriteErrDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_StatWriteErrDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_StatWriteErrDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_StatWriteErrDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_StatWriteErrDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;
                case IDCANCEL:
                    if (!BLD_StatWriteErrDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;
                default:
                    return BLD_StatWriteErrDlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
            return BLD_StatWriteErrDlgDefault(hDlg,message,wParam,lParam);
            break;
    }
    return TRUE; // Did process the message
}
```



```
//init.c
//created 12/3/93 by Laura Downey, Computer Scientist, NIST
//utility program to initialize aggregate statistics structure
//with all zeroes and to write the structure to the binary file stat#.fil
//compiled using MSC 6.0 pwb

//NOTE: init.exe needs to be located in the same directory as stat#.fil

#include <stdio.h>           //standard i/o

#include "struct.h"         //structure definitions, including ex_stat

struct ex_stat aggregate;  //by globally defining, aggregate is initialized
                           //to all zeroes

main()
{
    char STATFILE[15];      //holds file name entered by user

    FILE *sf;               //file pointer to file name entered by user

    //clear screen and prompt user to enter file name
    system("cls");
    printf("\n\nPLEASE ENTER THE NAME OF THE FILE CONTAINING THE STATISTICS (stat#.fil)\n\n");
    scanf("%s", STATFILE);

    //write aggregate statistics structure containing all zeroes to stat#.fil
    sf = fopen(STATFILE, "wb");
    if (sf == NULL)
        printf("\n\nUNABLE TO OPEN %s\n", STATFILE);
    else
    {
        fwrite(&aggregate, sizeof(struct ex_stat), 1, sf);
        fclose(sf);
    }

} //END OF MAIN
```

```
// Filename: PRINT.C
// "EAMAT42" Generated by WindowsMAKER Professional.
// Author: Laura L. Downey
```

```
#include "WINDOWS.H"
#include "GENERIC.H"
```

```
WMPDEBUG
#include "PRINT.WMC"
```

```
/* *****
/* Abort Procedure that catches printer problems */
/* *****
```

```
BOOL FAR PASCAL AbortProc(HDC hdc, short nCode)
{
    MSG msg ;

    while (PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg);
    }
    return TRUE ;
} //end of AbortProc
```

```
/* *****
/* displays "blanket report now printing" message */
/* *****
// *****
// Modal Dialog Box: PBLANKET
// *****
```

```
// Startup procedure for modal dialog box
int BLD_PrintBlanketDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_PrintBlanketDlgFuncDef(hWnd,(char *)NULL);
}
```

```
// Modal dialog box procedure
BOOL CALLBACK BLD_PrintBlanketDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    char PTitle[800];                /** title page string for blanket report **/

    FARPROC lpfnAbortProc;          //pointer to message function AbortProc

    HDC hdc;                        /** device context to printer **/

    int height;                     /** height in pixels of employee detail record **/
    int index;                      /** index of list box entry **/
    int perpage;                    /** # of employee records printed per page **/

    RECT rect;                      /** structure defining dimensions of employee
                                    record in pixels for printing **/

    WORD wId;

    switch(message)
    {
```

```
case WM_INITDIALOG:
    return BLD_PrintBlanketDlgDefault(hDlg,message,wParam,lParam);
    break;

case WM_COMMAND:
    wId=LOWORD(wParam);
    switch(wId)
    {
    case IDOK:
        if (!BLD_PrintBlanketDlgDefault(hDlg,message,wParam,lParam))
            EndDialog(hDlg, IDOK);
        break;

    case IDCONT:
        rect.left = 0;
        rect.top = 500;
        rect.right = 200;
        rect.bottom = 700;
        hdc = GetPrinterDC();

        EnableWindow(hDlg, FALSE); //disable window

        //start abort message function for protection
        lpfnAbortProc = MakeProcInstance ((FARPROC) AbortProc, hInst);
        Escape(hdc, SETABORTPROC, 0, (LPSTR) lpfnAbortProc, NULL);

        Escape(hdc, STARTDOC, strlen(PEDetailString)+1, (LPSTR) PEDetailString,
            (LPSTR) NULL);
        CreateTitlePage(PTitle);
        DrawText(hdc, (LPSTR) PTitle, -1, &rect, DT_NOCLIP | DT_EXTERNALLEADING
            | DT_EXPANDTABS);
        Escape(hdc, NEWFRAME, 0, 0L, 0L);
        rect.left = 0;
        rect.top = 0;
        rect.right = 0;
        rect.bottom = 0;
        perpage=0;
        for(index=0; index<=29; index++)
        {
            CopyBlanket_EDetail(index);
            CreateEDetail();
            height = DrawText(hdc, (LPSTR) PEDetailString, -1, &rect, DT_EXPANDTABS |
                DT_NOCLIP | DT_EXTERNALLEADING | DT_CALCRECT);
            DrawText(hdc, (LPSTR) PEDetailString, -1, &rect, DT_EXPANDTABS |
                DT_NOCLIP | DT_EXTERNALLEADING);
            rect.top = rect.top + height;
            perpage++;
            if (perpage > 2)
            {
                Escape(hdc, NEWFRAME, 0, 0L, 0L);
                perpage = 0;
                rect.top = 0;
            }
            else
            {
                height = DrawText(hdc, (LPSTR) separatorstring, -1, &rect, DT_NOCLIP |
                    DT_EXTERNALLEADING | DT_CALCRECT);
                DrawText(hdc, (LPSTR) separatorstring, -1, &rect, DT_NOCLIP |
                    DT_EXTERNALLEADING);
                rect.top = rect.top + height;
            }
        }
        Escape(hdc, NEWFRAME, 0, 0L, 0L);
        Escape(hdc, ENDDOC, 0, 0L, 0L);
        //finish print job
```



```

        FreeProcInstance (lpfnAbortProc);           //free abort message function
        EnableWindow(hDlg, TRUE);                   //enable current dialog
        DeleteDC(hdc);                               //delete device context
        EndDialog(hDlg, IDCNT);                     //end current dialog
        break; //end of IDCNT

    case IDCANCEL:
        if (!BLD_PrintBlanketDlgDefault(hDlg,message,wParam,lParam))
            EndDialog(hDlg, IDCANCEL);
        break;

    default:
        return BLD_PrintBlanketDlgDefault(hDlg,message,wParam,lParam);
        break;
    }
    break;

default:
    return BLD_PrintBlanketDlgDefault(hDlg,message,wParam,lParam);
    break;
}

return TRUE; // Did process the message
} //END OF BLANKET REPORT PRINTING MESSAGE DIALOG

/*****
/* displays "employee detail now printing" message */
/*****
// ****
// Modal Dialog Box: PEDETAIL
// ****

// Startup procedure for modal dialog box
int BLD_PrintEmpDetailDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_PrintEmpDetailDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_PrintEmpDetailDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    FARPROC lpfnAbortProc;    /** pointer to message function AbortProc **/

    HDC hdc;                  /** device context to printer **/

    int i;                    /** loop counter **/
    int num;                  /** number of copies to be printed **/

    RECT rect;                /** holds dimensions in pixels of
                                employee detail to be printed **/

    WORD        wId;

    switch(message)
    {
    case WM_INITDIALOG:
        return BLD_PrintEmpDetailDlgDefault(hDlg,message,wParam,lParam);
        break;

    case WM_COMMAND:
        wId=LOWORD(wParam);

```

```
switch(wId)
{
case IDOK:
    if (!BLD_PrintEmpDetailDlgDefault(hDlg,message,wParam,lParam))
        EndDialog(hDlg,IDOK);
    break;

case IDCNT:
    num = atoi(cnum);
    rect.left = 0;
    rect.top = 20;
    rect.right = 400;
    rect.bottom = 1000;
    hdc = GetPrinterDC();

    EnableWindow(hDlg, FALSE); //disable the window

    //start abort message function for protection
    lpfnAbortProc = MakeProcInstance ((FARPROC) AbortProc, hInst);
    Escape(hdc, SETABORTPROC, 0, (LPSTR) lpfnAbortProc, NULL);

    Escape(hdc, STARTDOC, strlen(PrintEDetail)+1, (LPSTR)PrintEDetail,
        (LPSTR) NULL);
    for(i = 1; i <= num; i++)
    {
        DrawText(hdc, (LPSTR)PrintEDetail, -1, &rect, DT_EXPANDTABS |
            DT_NOCLIP | DT_EXTERNALLEADING);
        Escape(hdc, NEWFRAME, 0, 0L, 0L);
    }
    Escape(hdc, ENDDOC, 0, 0L, 0L); //finish print job
    FreeProcInstance (lpfnAbortProc); //free abort message function
    EnableWindow(hDlg, TRUE); //enable current dialog
    DeleteDC(hdc); //destroy device context
    EndDialog(hDlg, IDCNT); //end dialog
    break;

case IDCANCEL:
    if (!BLD_PrintEmpDetailDlgDefault(hDlg,message,wParam,lParam))
        EndDialog(hDlg, IDCANCEL);
    break;

default:
    return BLD_PrintEmpDetailDlgDefault(hDlg,message,wParam,lParam);
    break;
}
break;

default:
    return BLD_PrintEmpDetailDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE; // Did process the message
} //END OF EMPLOYEE DETAIL PRINTING MESSAGE DIALOG
```

```
/* *****
/* displays "header detail now printing" message */
/* *****
// *****
// Modal Dialog Box: PHDETAIL
// *****
```

```
// Startup procedure for modal dialog box
int BLD_PrintHeaderDetailDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_PrintHeaderDetailDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_PrintHeaderDetailDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    FARPROC lpfnAbortProc;           //pointer to message function AbortProc

    HDC hdc;                          /** device context to printer **/

    RECT rect;                        /** holds dimensions in pixels of
                                     header detail to be printed **/

    WORD      wId;

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_PrintHeaderDetailDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    if (!BLD_PrintHeaderDetailDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;

                case IDCNT:
                    CreateHDetailString();
                    rect.left = 0;
                    rect.top = 20;
                    rect.right = 200;
                    rect.bottom = 200;
                    hdc = GetPrinterDC();

                    EnableWindow(hDlg, FALSE);           //disable window

                    //start abort message function for protection
                    lpfnAbortProc = MakeProcInstance ((FARPROC) AbortProc, hInst);
                    Escape(hdc, SETABORTPROC, 0, (LPSTR) lpfnAbortProc, NULL);

                    Escape(hdc, STARTDOC, strlen(PHDetailString)+1, (LPSTR) PHDetailString,
                        (LPSTR) NULL);
                    DrawText(hdc, (LPSTR) PHDetailString, -1, &rect, DT_EXPANDTABS |
                        DT_NOCLIP | DT_EXTERNALLEADING);
                    Escape(hdc, NEWFRAME, 0, 0L, 0L);
                    Escape(hdc, ENDDOC, 0, 0L, 0L);
                    FreeProcInstance (lpfnAbortProc);    //free abort message function
                    EnableWindow(hDlg, TRUE);           //enable current dialog
                    DeleteDC(hdc);                       //destroy device context
                    EndDialog(hDlg, IDCNT);             //end dialog
                    break; //end of IDCNT

                case IDCANCEL:
                    if (!BLD_PrintHeaderDetailDlgDefault(hDlg,message,wParam,lParam))
```



```
        EndDialog(hDlg, IDCANCEL);
    break;

    default:
        return BLD_PrintHeaderDetailDlgDefault(hDlg, message, wParam, lParam);
    break;
}
break;

default:
    return BLD_PrintHeaderDetailDlgDefault(hDlg, message, wParam, lParam);
break;
}
return TRUE; // Did process the message
} //END OF HEADER DETAIL PRINTING MESSAGE DIALOG


/*****
/* displays "report now printing" message */
/*****
// ****
//          Modal Dialog Box: PINFO
// ****

// Startup procedure for modal dialog box
int BLD_Function5DlgFunc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    return BLD_Function5DlgFuncDef(hWnd, (char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_Function5DlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
    case WM_INITDIALOG:
        return BLD_Function5DlgDefault(hDlg, message, wParam, lParam);
        break;

    case WM_COMMAND:
        wId=LOWORD(wParam);
        switch(wId)
        {
        case IDOK:
            if (!BLD_Function5DlgDefault(hDlg, message, wParam, lParam))
                EndDialog(hDlg, IDOK);
            break;

        case IDCANCEL:
            if (!BLD_Function5DlgDefault(hDlg, message, wParam, lParam))
                EndDialog(hDlg, IDCANCEL);
            break;

        default:
            return BLD_Function5DlgDefault(hDlg, message, wParam, lParam);
            break;
        }
    }
    break;
}
```

```
default:
    return BLD_Function5DlgDefault (hDlg,message,wParam,lParam);
    break;
}
return TRUE;// Did process the message
} //END OF REPORT NOW PRINTING DIALOG

/*****
/* displays "totals now printing" message */
*****/
// *****
//          Modal Dialog Box: PTOTALS
// *****

// Startup procedure for modal dialog box
int BLD_TotNowPrintDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_TotNowPrintDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_TotNowPrintDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    FARPROC lpfnAbortProc;    //pointer to message function AbortProc

    HDC hdc;                  /** device context to printer **/

    RECT rect;                /** holds dimensions in pixels of
                                header detail to be printed **/

    WORD        wId;

    switch(message)
    {
    case WM_INITDIALOG:
        return BLD_TotNowPrintDlgDefault (hDlg,message,wParam,lParam);
        break;

    case WM_COMMAND:
        wId=LOWORD(wParam);
        switch(wId)
        {
        case IDOK:
            if (!BLD_TotNowPrintDlgDefault (hDlg,message,wParam,lParam))
                EndDialog(hDlg,IDOK);
            break;

        case IDCNT:
            rect.left = 0;
            rect.top = 200;
            rect.right = 1000;
            rect.bottom = 1500;
            CreatePrintTotString();
            hdc = GetPrinterDC();

            EnableWindow(hDlg, FALSE);                //disable window

            //start abort message function for protection
            lpfnAbortProc = MakeProcInstance ((FARPROC) AbortProc, hInst);
            Escape(hdc, SETABORTPROC, 0, (LPSTR) lpfnAbortProc, NULL);
        }
    }
}
```

```

        Escape(hdc, STARTDOC, strlen(PrintTotString)+1, (LPSTR)PrintTotString,
            (LPSTR)NULL);
        DrawText(hdc, (LPSTR)PrintTotString, -1, &rect, DT_NOCLIP |
            DT_EXTERNALLEADING | DT_EXPANDTABS);
        Escape(hdc, NEWFRAME, 0, 0L, 0L);
        Escape(hdc, ENDDOC, 0, 0L, 0L);

        FreeProcInstance (lpfnAbortProc);           //free abort message function
        EnableWindow(hDlg, TRUE);                   //enable current dialog
        DeleteDC(hdc);                               //destroy device context
        EndDialog(hDlg, IDCNT);                      //end dialog
        break; //end of IDCNT

    case IDCANCEL:
        if (!BLD_TotNowPrintDlgDefault(hDlg, message, wParam, lParam))
            EndDialog(hDlg, IDCANCEL);
        break;

    default:
        return BLD_TotNowPrintDlgDefault(hDlg, message, wParam, lParam);
        break;
    }
    break;

default:
    return BLD_TotNowPrintDlgDefault(hDlg, message, wParam, lParam);
    break;
}

return TRUE; // Did process the message
} //END OF REPORT TOTALS PRINTING MESSAGE DIALOG

/*****
/* prompts user to enter the number of employee details to print*/
*****/
// ****
//      Modal Dialog Box: NCOPY
// ****

// Startup procedure for modal dialog box
int BLD_GetNumCopyDlgFunc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    return BLD_GetNumCopyDlgFuncDef(hWnd, (char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_GetNumCopyDlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            SetDlgItemText(hDlg, IDMF_NumCopies, "1");
            return BLD_GetNumCopyDlgDefault(hDlg, message, wParam, lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:

```



```
        GetDlgItemText(hDlg, IDMF_NumCopies, cnum, 6); /** get N+1 char **/  
        EndDialog(hDlg, IDOK);  
        break;  
  
    case IDCANCEL:  
        if (!BLD_GetNumCopyDlgDefault(hDlg, message, wParam, lParam))  
            EndDialog(hDlg, IDCANCEL);  
        break;  
  
    default:  
        return BLD_GetNumCopyDlgDefault(hDlg, message, wParam, lParam);  
        break;  
    }  
    break;  
  
default:  
    return BLD_GetNumCopyDlgDefault(hDlg, message, wParam, lParam);  
    break;  
}  
return TRUE; // Did process the message  
} //END OF "HOW MANY" EMPLOYEE DETAILS TO PRINT ENTRY SCREEN
```

```
/** added 7/13/92 - function to create device context for printer **/  
/***** GETPRINTERDC *****/  
/** this function gets the device context to the default printer **/  
HDC GetPrinterDC()  
{  
    char szPrinter [80];  
    char *szDevice, *szDriver, *szOutput;  
  
    GetProfileString("windows", "device", ".,.", szPrinter, 80);  
  
    if ((szDevice = strtok(szPrinter, ",")) &&  
        (szDriver = strtok(NULL, ",")) &&  
        (szOutput = strtok(NULL, ",")))  
        return CreateDC(szDriver, szDevice, szOutput, NULL);  
  
    return 0;  
} //END OF GETPRINTERDC
```

```
// Filename: SERVICE.C
// "EAMAT42" Generated by WindowsMAKER Professional.
// Author: Laura L. Downey

//
// *****
// Code in this file is initially generated by WindowsMAKER Professional.
// This file contains functions you can change
// to provide whatever functionality you require.
//
// All actual processing are done in the .WMC file. To override that
// functionality add your own code in these functions.
//
// For more information,
// see the section "How code is generated" in the documentation.
//
// *****
//

#include "WINDOWS.H"
#include "GENERIC.H"

WMPDEBUG
#include "SERVICE.WMC"

// *****
// ERROR MESSAGE HANDLING
// *****

int BLDDisplayMessage(HWND hWnd,UINT uMsg,char *pContext,int iType)
{
    return BLDDisplayMessageDef(hWnd,uMsg,pContext,iType);
}

// *****
// FUNCTIONS FOR DRAWING GRAPHIC BUTTONS
// *****

BOOL BLDBitmapToScreen(HDC hDestDC, char *pBitmapName,
                      int X,int Y,int nWidth,int nHeight,
                      DWORD dwRop,BOOL bStretch)
{
    return BLDBitmapToScreenDef(hDestDC,pBitmapName,X,Y,nWidth,nHeight,
                                dwRop,bStretch);
}

BOOL BLDDrawIcon(LPDRAWITEMSTRUCT lpDrawItem,char *pIconName)
{
    return BLDDrawIconDef(lpDrawItem,pIconName);
}

BOOL BLDDrawBitmap(LPDRAWITEMSTRUCT lpDrawItem,char *pBitmapName,BOOL bStretch)
{
    return BLDDrawBitmapDef(lpDrawItem,pBitmapName,bStretch);
}

BOOL BLDDrawBkgndIcon(HWND hDlg,PAINTSTRUCT *pPs,char *pIconName,int iCtrlID)
{

```

```
    return BLDDrawBkgndIconDef(hDlg,pPs,pIconName,iCtrlID);
}

BOOL BLDDrawBkgndBitmap(HWND hDlg,PAINTSTRUCT *pPs,char *pBitmapName,
    int iCtrlID,BOOL bStretch,BOOL bCtrl)
{
    return BLDDrawBkgndBitmapDef(hDlg,pPs,pBitmapName,iCtrlID,bStretch,bCtrl);
}

BOOL BLDDrawAutoState(LPDRAWITEMSTRUCT lpDrawItem,char *szResource,BOOL bBitmap,BOOL bStretch)
{
    return BLDDrawAutoStateDef(lpDrawItem, szResource, bBitmap, bStretch);
}

BOOL BLDDrawStateBitmap(LPDRAWITEMSTRUCT lpDrawItem,char *szNormal,char *szFocus,
    char *szSelected,char *szDisabled,BOOL bStretch)
{
    return BLDDrawStateBitmapDef(lpDrawItem,szNormal,szFocus,
        szSelected,szDisabled,bStretch);
}

BOOL BLDDrawStateIcon(LPDRAWITEMSTRUCT lpDrawItem,char *szNormal,char *szFocus,
    char *szSelected,char *szDisabled)
{
    return BLDDrawStateIconDef(lpDrawItem,szNormal,szFocus,
        szSelected,szDisabled);
}

BOOL BLDDrawItem(HWND hDlg,LPDRAWITEMSTRUCT lpDrawItem)
{
    return BLDDrawItemDef(hDlg,lpDrawItem);
}

// *****
//          FUNCTION FOR LOADING BITMAP
// *****

HBITMAP BLDLoadBitmap(HINSTANCE hInstance,char *pBitmapName)
{
    return LoadBitmap(hInstance,(LPSTR)pBitmapName);
}

// *****
//          FUNCTIONS FOR DIALOG BOX SCROLLING
// *****

void BLDGetDlgScrolled(HWND hDlg,int *pxScrolled,int *pyScrolled)
{
    BLDGetDlgScrolledDef(hDlg,pxScrolled,pyScrolled);
}

void BLDSetDlgScrolled(HWND hDlg,int xScrolled,int yScrolled)
{
    BLDSetDlgScrolledDef(hDlg,xScrolled,yScrolled);
}
```



```
BOOL BLDExitScrollDlg(HWND hDlg)
{
    return BLDExitScrollDlgDef(hDlg);
}

void BLDFindCtrlsRightBottom(HWND hDlg,int *xRight,int *yBottom)
{
    BLDFindCtrlsRightBottomDef(hDlg,xRight,yBottom);
}

void BLDCalcScrollRanges(HWND hDlg,int *xRange,int *yRange,int xScrolled,
                          int yScrolled,int iRightOf,int iBelow)
{
    BLDCalcScrollRangesDef(hDlg,xRange,yRange,xScrolled,yScrolled,iRightOf,iBelow);
}

BOOL BLDScrollDlg(HWND hDlg,UINT message,int nScrlCode,int nPos,int iVertLine,
                  int iHorLine,int iVertPage,int iHorPage,int iRightOf,
                  int iBelow,BOOL bInvalidate)
{
    return BLDScrollDlgDef(hDlg, message,nScrlCode,nPos,iVertLine,iHorLine,
                           iVertPage,iHorPage,iRightOf,iBelow,bInvalidate);
}

// *****
//      FUNCTION FOR CREATING CONTROLS IN MAIN WINDOW
// *****

HWND BLDCreateClientControls(char *pTemplateName,DLGPROC lpNew)
{
    return BLDCreateClientControlsDef(pTemplateName,lpNew);
}

void BLDMoveWindow(HWND hWnd,int x,int y,int nWidth,int nHeight,BOOL bRepaint)
{
    BLDMoveWindowDef(hWnd,x,y,nWidth,nHeight,bRepaint);
}

void BLDMoveDlgClient(HWND ParenthWnd,HWND hNew)
{
    BLDMoveDlgClientDef(ParenthWnd,hNew);
}

void BLDSetClientFocus(HWND hWnd)
{
    BLDSetClientFocusDef(hWnd);
}

void BLDClientMove(HWND hWnd)
{
    BLDClientMoveDef(hWnd);
}
```

4

```

BOOL BLDInitCtrlFont(HWND hDlg,int iCtrlId,int nHeight,int nWidth,int nEscapement,
    int nOrientation,int fnWeight,BYTE fbItalic,BYTE fbUnderline,
    BYTE fbStrikeOut,BYTE fbCharSet,BYTE fbOutputPrecision,
    BYTE fbClipPrecision,BYTE fbQuality,BYTE fbPitchAndFamily,
    char *lpszFace)
{
    return BLDInitCtrlFontDef(hDlg,iCtrlId,nHeight,nWidth,nEscapement,
        nOrientation,fnWeight,fbItalic,fbUnderline,
        fbStrikeOut,fbCharSet,fbOutputPrecision,
        fbClipPrecision,fbQuality,
        fbPitchAndFamily,lpszFace);
}

```

```
BOOL BLDExitCtrlFont(HWND hDlg,int iCtrlId)
{
    return BLDExitCtrlFontDef(hDlg,iCtrlId);
}

// *****
//      FUNCTIONS DIALOG BOX AND CONTROLS BACKGROUND
// *****

HBRUSH BLDCtlColorBrushSetOrg(HWND hWnd,HDC hDC)
{
    return BLDCtlColorBrushSetOrgDef(hWnd,hDC);
}

BOOL BLDInitSolidBrush(HWND hWnd,COLORREF ColorRef)
{
    return BLDInitSolidBrushDef(hWnd,ColorRef);
}

BOOL BLDInitPatternBrush(HWND hWnd,char *pBitmapName)
{
    return BLDInitPatternBrushDef(hWnd,pBitmapName);
}

BOOL BLDExitBrush(HWND hWnd)
{
    return BLDExitBrushDef(hWnd);
}

HBRUSH BLDCtlColorStockBrush(HWND hWnd,int fnObject)
{
    return BLDCtlColorStockBrushDef(hWnd,fnObject);
}

HBRUSH BLDCtlColorPropBrush(HWND hWnd)
{
    return BLDCtlColorPropBrushDef(hWnd);
}

HBRUSH BLDCtlColorDefaultBrush(HWND hWnd)
{
    return BLDCtlColorDefaultBrushDef(hWnd);
}

// *****
//      FUNCTIONS FOR HELP HANDLING
// *****

BOOL BLDCheckF1HelpKey(BOOL bShift)
{
    return BLDCheckF1HelpKeyDef(bShift);
}

void BLDHelpTranslation(MSG *pmsg)
```



```
{
    BLDHelpTranslationDef(pmsg);
}

void BLDShowHelp(HWND hWnd,UINT fuCommand,DWORD dwData)
{
    BLDShowHelpDef(hWnd,fuCommand,dwData);
}

BOOL BLDHelpFilter(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam,DWORD dwHelpId,
    LPARAM *plRetval,BOOL bFromDlg)
{
    return BLDHelpFilterDef(hWnd,message,wParam,lParam,dwHelpId,plRetval,bFromDlg);
}

LRESULT BLDDefWindowProcMsg(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLDDefWindowProcMsgDef(hWnd,message,wParam,lParam);
}

// *****
//          FUNCTIONS FOR HANDLING OF
//          MULTIPLE INSTANCES OF TOOLBARS AND
//          CLIENT AREA CONTROLS
// *****

BOOL BLDAddClientDlg(HWND hDlg,DLGPROC lpProc)
{
    return BLDAddClientDlgDef(hDlg,lpProc);
}

BOOL BLDRemoveClientDlg(HWND hDlg)
{
    return BLDRemoveClientDlgDef(hDlg);
}

BOOL BLDIsClientDlgDialogMessage(MSG *pMsg)
{
    return BLDIsClientDlgDialogMessageDef(pMsg);
}
```

[illegible]

```
        aggregate.rs_tot_time / (double)aggregate.rtot_non_interrupt;
    aggregate.rq_avg_time =
        aggregate.rq_tot_time / (double)aggregate.rtot_non_interrupt;
    }

    break;
case YES://interrupted
    //totals
    aggregate.rsi_tot_time = aggregate.rsi_tot_time + current.snet_time;
    aggregate.rqi_tot_time = aggregate.rqi_tot_time + current.qnet_time;
    aggregate.rtot_interrupt++;

    //averages
    if (aggregate.rtot_interrupt != 0);
    {
        aggregate.rsi_avg_time =
            aggregate.rsi_tot_time / (double)aggregate.rtot_interrupt;
        aggregate.rqi_avg_time =
            aggregate.rqi_tot_time / (double)aggregate.rtot_interrupt;
    }

    break;
} //end of switch current.interrupted

//addt'l matches tally for resolved cases
switch (current.add_match)
{
    case 0:
        aggregate.rtot_add_match0++;
        break;
    case 1:
        aggregate.rtot_add_match1++;
        aggregate.rtot_add_match++;
        break;
    case 2:
        aggregate.rtot_add_match2++;
        aggregate.rtot_add_match = aggregate.rtot_add_match + current.add_match;
        break;
    case 3:
        aggregate.rtot_add_match3++;
        aggregate.rtot_add_match = aggregate.rtot_add_match + current.add_match;
        break;
    default:
        aggregate.rtot_add_match4_plus++;
        aggregate.rtot_add_match = aggregate.rtot_add_match + current.add_match;
        break;
} //end of switch current.add_match
break;
case NO: //unresolved
    switch (current.interrupted)
    {
        case NO: //not interrupted
            //totals
            aggregate.us_tot_time = aggregate.us_tot_time + current.snet_time;
            aggregate.uq_tot_time = aggregate.uq_tot_time + current.qnet_time;
            aggregate.utot_non_interrupt++;

            //averages
            if (aggregate.utot_non_interrupt != 0);
            {
                aggregate.us_avg_time =
                    aggregate.us_tot_time / (double)aggregate.utot_non_interrupt;
                aggregate.uq_avg_time =
                    aggregate.uq_tot_time / (double)aggregate.utot_non_interrupt;
```



```
    }

    break;
case YES://interrupted
    //totals
    aggregate.usi_tot_time = aggregate.usi_tot_time + current.snet_time;
    aggregate.uqi_tot_time = aggregate.uqi_tot_time + current.qnet_time;
    aggregate.utot_interrupt++;

    //averages
    if (aggregate.utot_interrupt != 0)
    {
        aggregate.usi_avg_time =
            aggregate.usi_tot_time / (double)aggregate.utot_interrupt;
        aggregate.uqi_avg_time =
            aggregate.uqi_tot_time / (double)aggregate.utot_interrupt;
    }

    break;
} //end of switch current.interrupted

//addt'l matches tally for unresolved cases
switch (current.add_match)
{
    case 0:
        aggregate.utot_add_match0++;
        break;
    case 1:
        aggregate.utot_add_match1++;
        aggregate.utot_add_match++;
        break;
    case 2:
        aggregate.utot_add_match2++;
        aggregate.utot_add_match = aggregate.utot_add_match + current.add_match;
        break;
    case 3:
        aggregate.utot_add_match3++;
        aggregate.utot_add_match = aggregate.utot_add_match + current.add_match;
        break;
    default:
        aggregate.utot_add_match4_plus++;
        aggregate.utot_add_match = aggregate.utot_add_match + current.add_match;
        break;
} //end of switch current.add_match
break;
} //end of switch on current.resolved

//
//MORE TOTALS
//
//total resolved cases
aggregate.tot_resolved =
    aggregate.rtot_non_interrupt + aggregate.rtot_interrupt;

//total unresolved cases
aggregate.tot_unresolved =
    aggregate.utot_non_interrupt + aggregate.utot_interrupt;

//total cases without interruptions
aggregate.tot_non_interrupt =
    aggregate.rtot_non_interrupt + aggregate.utot_non_interrupt;
```

```
//total cases with interruptions
aggregate.tot_interrupt =
    aggregate.rtot_interrupt + aggregate.utot_interrupt;

//total cases
aggregate.tot_cases =
    aggregate.tot_resolved + aggregate.tot_unresolved;

//
//MORE AVERAGES
//
//average # of additional matches per resolved case
if (aggregate.tot_resolved != 0)
    aggregate.ravg_add_match =
        (double)aggregate.rtot_add_match / (double)aggregate.tot_resolved;

//average # of additional matches per unresolved case
if (aggregate.tot_unresolved != 0)
    aggregate.uavg_add_match =
        (double)aggregate.utot_add_match / (double)aggregate.tot_unresolved;

//average # of additional matches per case
if (aggregate.tot_cases != 0)
    aggregate.avg_add_match =
        (double)aggregate.tot_add_matches / (double)aggregate.tot_cases;

//PERCENTAGES

//
// resolved cases
//

if (aggregate.tot_resolved != 0)
{
    // % 0 addt'l matches
    aggregate.rper_add_match0 =
        ((double)aggregate.rtot_add_match0 / (double)aggregate.tot_resolved) * 100;

    // % 1 addt'l match
    aggregate.rper_add_match1 =
        ((double)aggregate.rtot_add_match1 / (double)aggregate.tot_resolved) * 100;

    // % 2 addt'l matches
    aggregate.rper_add_match2 =
        ((double)aggregate.rtot_add_match2 / (double)aggregate.tot_resolved) * 100;

    // % 3 addt'l matches
    aggregate.rper_add_match3 =
        ((double)aggregate.rtot_add_match3 / (double)aggregate.tot_resolved) * 100;

    // % 4 or more addt'l matches
    aggregate.rper_add_match4_plus =
        ((double)aggregate.rtot_add_match4_plus / (double)aggregate.tot_resolved) * 100;
}

//
// unresolved cases
//
```

0;

```
    if (aggregate.tot_unresolved != 0)
    {
        /*% 0 addt'l matches
        aggregate.uper_add_match0 =
            ((double)aggregate.utot_add_match0 / (double)aggregate.tot_unresolved) * 100;

        /*% 1 addt'l match
        aggregate.uper_add_match1 =
            ((double)aggregate.utot_add_match1 / (double)aggregate.tot_unresolved) * 100;

        /*% 2 addt'l matches
        aggregate.uper_add_match2 =
            ((double)aggregate.utot_add_match2 / (double)aggregate.tot_unresolved) * 100;

        /*% 3 addt'l matches
        aggregate.uper_add_match3 =
            ((double)aggregate.utot_add_match3 / (double)aggregate.tot_unresolved) * 100;

        /*% 4 or more addt'l matches
        aggregate.uper_add_match4_plus =
            ((double)aggregate.utot_add_match4_plus / (double)aggregate.tot_unresolved) *
100;
    }

    break;

} //end of switch on current.outlier
} //end of complete single query case

} //END OF FILL_AGGREGATE()
```

```
/****** FILL_CURRENT *****/
void fill_current() //calculates current times and assigns
                  //outlier status if appropriate
```

```
#define HALF_HOUR 1800 //number of seconds in a half hour
```

```
{
    if (current.stop_single != 0)
    {
        current.snet_time = difftime(current.stop_single, current.start_single);
        current.qnet_time = difftime(current.stop_single, current.start_qinfo);
        if (current.snet_time > HALF_HOUR)
            current.outlier = YES;
    }

    if (current.stop_browse != 0)
        current.bnet_time = difftime(current.stop_browse, current.start_browse);
} //END OF FILL_CURRENT()
```

```
/****** INIT_AGGREGATE *****/
void init_aggregate(hDlg,message,wParam,lParam) //initializes aggregate ex_stat structure
                                              //with current info from stat#.fil
```

```
HWND hDlg;
UINT message;
WPARAM wParam;
LONG lParam;
```

```
{
```





```
HWND hDlg;
UINT message;
WPARAM wParam;
LONG lParam;

{
    char c_usernum[3];    //user number converted to character for concatenation
                          //in file name

    char STATFILE[14];    //stat#.fil

    FILE *sf;             //file pointer to stat#.fil

    int num;              //was anything read from the file


    //build file name using current user number

    strcpy(STATFILE, "stat");
    strcat(STATFILE, itoa(USER, c_usernum, 10) );
    strcat(STATFILE, ".fil");


    //read last set of aggregate statistics and report any errors
    sf = fopen(STATFILE, "wb");
    if (sf == NULL)
        BLD_StatWOpenDlgFunc(hDlg,message,wParam,lParam);
    else
    {
        num = fwrite(&aggregate, sizeof(struct ex_stat), 1, sf);
        if (num == 0)
            BLD_StatWriteErrDlgFunc(hDlg,message,wParam,lParam);
        fclose(sf);
    }
} //END OF WRITE_AGGREGATE()
```

```
/** November 22, 1993 */
/** Version 4.2 */
/**Filename: USERCODE.C */
/**"EAMATE" Generated by WindowsMAKER Professional */
/**Author: Laura L. Downey */

/** 11/15/93 - all changes, updates, suggestions incorporated */
/** usercode.c is ready for use */
/** only remaining code to be implemented is statistic collection */

// 11/22/93 - statistics code added

// 2/4/94 - RPC numbers changed, error codes adjusted, & file names adjusted

/** include files */
#include <WINDOWS.H> /** windows header file */
#include "GENERIC.H" /** master header file for the application */
#include <fcntl.h>
#include <io.h>
#include "USERCODE.WMC"
#include "global.h" /** includes global variables */
#include <c:\ptk40\include\rpc\rpc.h> /** added 1/5/93 by LLD per PC-NFS instructions */
/
#include "p_prot.h" /** per PC-NFS */
#include "p_clnt.h" /** per PC-NFS */
#include "tklib.h" /** per PC-NFS */

/** local file definitions */
#define EFILE "err.fil" /** error file for rpc errors */
#define UFILE "usernum.fil" /** user number stored here */

/** misc. definitions including
    rpc definitions */
#define ADD_MATCHES 0x37000000 /** additional matches identifier for rpc call */
#define BROWSE_REPORT 0x35000000 /** browse report identifier for rpc call */
#define DEF_SEQ_NO "AAA" /** default sequence number for employer report */
/
#define GET_BLANKET 0x32000000 /** blanket info identifier for rpc call */
#define GET_EMPL_DETAIL 0x34000000 /** employee detail identifier for rpc call */
#define GET_HEADER 0x30000000 /** all header info identifier for rpc call */
#define GET_SEQ_HEADER 0x36000000 /** specific header info identifier for rpc call */
*/
#define HOST "demeter" /** server name */
#define MAXBRLB 100 /** max increment of records to be added to
    browse list box */
#define MAXB 30 /** max number of records for blanket listbox */
#define MAXQLB 100 /** max increment of records to be added to
    query list box */
#define PASSWORD "scout" /** password for printing full report */
#define PRINT_REPORT 0x33000000 /** print report info identifier for rpc call */
#define PROCNUM 1 /** procedure number of rpc call */
#define RETRY_TIME 30 /** retry timeout for rpc calls */
#define TOT_TIME 30 /** total timeout for rpc calls */
#define SINGLE_QUERY 0x31000000 /** single-query info identifier for rpc call */
#define UDP "udp" /** protocol used with creating client handle */
#define VERSNUM 1 /** version number of rpc call */

struct ex_stat aggregate; /**statistics structure
struct in_stat current; /**statistics structure
```



```
//USER ENTRY SCREENS AND DISPLAY INFORMATION SCREENS
```

```
/******  
/* displays main window of application */  
/******
```

```
HWND BLD_MAINClFunc(HWND,message,wParam,lParam) /* Startup procedure for window in client  
area */
```

```
HWND hWnd;
```

```
UINT message;
```

```
WPARAM wParam;
```

```
LONG lParam;
```

```
{
```

```
    HWND hNew;
```

```
    FARPROC lpNew;
```

```
    lpNew = MakeProcInstance(BLD_MAINClProc,hInst);
```

```
    if (!(hNew = BLDCreateClientControls("MAIN",lpNew)))
```

```
    {
```

```
        FreeProcInstance(lpNew);
```

```
        BLDDisplayMessage(hWnd,BLD_CannotCreate,"MAIN",  
                           MB_OK | MB_ICONHAND);
```

```
    }
```

```
    return hNew;
```

```
}
```

```
/*Procedure for window in client area */
```

```
BOOL FAR PASCAL BLD_MAINClProc(HWND, message, wParam, lParam)
```

```
HWND hDlg;
```

```
UINT message;
```

```
WPARAM wParam;
```

```
LONG lParam;
```

```
{
```

```
    switch(message)
```

```
    {
```

```
        case WM_INITDIALOG:
```

```
            return BLD_MAINDlgDefault(hDlg,message,wParam,lParam);
```

```
            break;
```

```
        case WM_NCDESTROY:
```

```
            FreeProcInstance(lpClient);
```

```
            hClient = 0;
```

```
            break;
```

```
        case WM_COMMAND:
```

```
            switch(wParam)
```

```
            {
```

```
                case IDOK:
```

```
                    return BLD_MAINDlgDefault(hDlg,message,wParam,lParam);
```

```
                    break;
```

```
                case IDCANCEL:
```

```
                    return BLD_MAINDlgDefault(hDlg,message,wParam,lParam);
```

```
                    break;
```

```
            default:
```

```
        return BLD_MAINDlgDefault (hDlg,message,wParam,lParam);
        break;
    }
    break;

    default:
        return BLD_MAINDlgDefault (hDlg,message,wParam,lParam);
        break;
}
return TRUE; /* Processed the message */
} //END OF DISPLAY MAIN WINDOW


/*****
/* displays credits in main window */
*****/

int BLD_FunctionDlgFunc (hWnd,message,wParam,lParam) /* Startup procedure for modal dialog
box */
HWND hWnd;
UINT message;
WPARAM wParam;
LONG lParam;
{
    FARPROC lpProc;
    int ReturnValue;

    lpProc = MakeProcInstance (BLD_FunctionDlgProc,hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)"MAIN1", hWnd, lpProc);
    FreeProcInstance(lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage (hWnd,BLD_CannotCreate,"MAIN1",
                           MB_OK | MB_ICONHAND);
    return ReturnValue;
}

BOOL FAR PASCAL BLD_FunctionDlgProc(hDlg, message, wParam, lParam) /* Modal dialog box pro
cedure */
HWND hDlg;
UINT message;
WPARAM wParam;
LONG lParam;
{
    FILE *uf;           //file pointer to usernum.fil

    int num;           //data check for usernum.fil, was anything read from the file

    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_FunctionDlgDefault (hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            switch(wParam)
            {
                case ID_CONT:
                    EndDialog(hDlg,ID_CONT);
                    uf = fopen(UFILE, "rb");
                    if (uf == NULL)
```

```

        BLD_UserNumErrDlgFunc(hDlg,message,wParam,lParam);
    else
    {
        num = fscanf(uf, "%i", &USER);
        if (num == 0)
            BLD_UserNumErrDlgFunc(hDlg,message,wParam,lParam);
        fclose(uf);
    }

    //initialize aggregate statistic structure
    init_aggregate(hDlg,message,wParam,lParam);
    break;

case IDCANCEL:
    if (!BLD_FunctionDlgDefault(hDlg,message,wParam,lParam))
        EndDialog(hDlg,IDCANCEL);
    break;

default:
    return BLD_FunctionDlgDefault(hDlg,message,wParam,lParam);
    break;
}
break;

default:
    return BLD_FunctionDlgDefault(hDlg,message,wParam,lParam);
    break;
}

return TRUE; /* Did process the message */
} //END OF DISPLAY CREDITS IN MAIN WINDOW

```

```

/*****
**/
/* prompts user to enter search info for a particular employee
**/
/*****
**/

```

```

int BLD_QUERYDlgFunc(hWnd,message,wParam,lParam) /* Startup procedure for modal dialog box
*/
HWND hWnd;
UINT message;
WPARAM wParam;
LONG lParam;
{
    FARPROC lpProc;
    int ReturnValue;

    lpProc = MakeProcInstance(BLD_QUERYDlgProc,hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)"QUERY", hWnd, lpProc);
    FreeProcInstance(lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,"QUERY",
            MB_OK | MB_ICONHAND);

    return ReturnValue;
}

```

```

BOOL FAR PASCAL BLD_QUERYDlgProc(hDlg, message, wParam, lParam) /* Modal dialog box proced
ure */

```



```
HWND hDlg;
UINT message;
WPARAM wParam;
LONG lParam;
{
    char BRFILE[14],          //browse file
        DFILE[14],          //detail file
        HFILE[14],          //header file
        QFILE[13];          //query file

    char c_usernum[3];        //user number converted to character for concatenation
                                //in file name

    CLIENT _TKFAR *clnt;      /** handle to client **/

    FILE *hfil,*brfil;        /** pointers to server data files **/
    FILE *qf, *ef;            /** pointer to query file and error file **/

    HCURSOR hCur;            /** cursor handle **/

    HWND hEIN;                /** handle to IDMF_QEIN **/
    HWND hLName;              /** handle to IDMF_LName **/
    HWND hYear;               /** handle to IDMF_QYear **/

    int srpcres, sres, cres;   /** rpc call result, single query call result,
                                and clnt_control result **/
    int usernum, hrpcres, hres; /** user number, rpc call result, and header call result *
*/

    struct query hquery, squery; /** structure to hold header query parameters, & single qu
ery
                                parameters **/

    struct timeval timeout;     /** timevalue structure for clnt_control **/

    //build file names using current user number
    strcpy(BRFILE, "d:brow");
    strcat(BRFILE, itoa(USER, c_usernum, 10) );
    strcat(BRFILE, ".txt\0");

    strcpy(DFILE, "d:det1");
    strcat(DFILE, itoa(USER, c_usernum, 10) );
    strcat(DFILE, ".txt\0");

    strcpy(HFILE, "d:hdr");
    strcat(HFILE, itoa(USER, c_usernum, 10) );
    strcat(HFILE, ".txt\0");

    strcpy(QFILE, "d:query");
    strcat(QFILE, itoa(USER, c_usernum, 10) );
    strcat(QFILE, ".txt\0");

    switch(message)
    {
        case WM_INITDIALOG:
            //initialize current statistic structure
            init_current();

            //record statistics
            current.start_single = time(NULL); //store start single query time
            aggregate.tot_single_query++;      //increment # of single query operations c
```

hosen

```
return BLD_QUERYDlgDefault (hDlg,message,wParam,lParam);
```

```
break;
```

```
case WM_COMMAND:
```

```
switch(wParam)
```

```
{
```

```
case IDOK:
```

```
/** get info entered by user **/
```

```
QYear[0] = 0;
```

```
QEIN[0] = 0;
```

```
QEstab[0] = 0;
```

```
LName[0] = 0;
```

```
FName[0] = 0;
```

```
QSSN[0] = 0;
```

```
GetDlgItemText (hDlg, IDMF_QYear, (LPSTR)QYear, 5); /** get N+1 characters **/
```

```
GetDlgItemText (hDlg, IDMF_QEIN, (LPSTR)QEIN, 11);
```

```
GetDlgItemText (hDlg, IDMF_QEstab, (LPSTR)QEstab, 5);
```

```
GetDlgItemText (hDlg, IDMF_LName, (LPSTR)LName, 16);
```

```
GetDlgItemText (hDlg, IDMF_FName, (LPSTR)FName, 13);
```

```
GetDlgItemText (hDlg, IDMF_QSSN, (LPSTR)QSSN, 12);
```

```
if (strcmp(QYear, "1991") != 0)
```

```
/** validate the year field
```

```
**/
```

```
{
```

```
BLD_Year_ErrDlgFunc (hDlg,message,wParam,lParam);
```

```
hYear = GetDlgItem (hDlg, IDMF_QYear);
```

```
SetFocus (hYear);
```

```
}
```

```
else
```

```
//year OK, check name if entered
```

```
{
```

```
hCur = SetCursor (LoadCursor (NULL, IDC_WAIT)); //change to hourglass
```

```
ShowCursor (TRUE); //show hourglass
```

```
if (((strlen(LName) == 0) && (strlen(FName) != 0)) || ((strlen(LName) != 0)
```

```
&&
```

```
(strlen(FName) == 0)))
```

```
{
```

```
/** if a name is entered, both first and last name must be entered **/
```

```
BLD_NMSGDlgFunc (hDlg,message,wParam,lParam);
```

```
ShowCursor (FALSE);
```

```
//hide hourglass
```

```
SetCursor (hCur);
```

```
//reset to arrow
```

```
hLName = GetDlgItem (hDlg, IDMF_LName);
```

```
SetFocus (hLName);
```

```
}
```

```
else
```

```
//name OK, check SSN if entered
```

```
if ((strlen(LName) == 0) && (strlen(QSSN) < 11))
```

```
{
```

```
/** either both first and last name must be entered or SSN **/
```

```
BLD_QueryMessageDlgFunc (hDlg,message,wParam,lParam);
```

```
ShowCursor (FALSE);
```

```
//hide hourglass
```

```
SetCursor (hCur);
```

```
//reset to arrow
```

```
hLName = GetDlgItem (hDlg, IDMF_LName);
```

```
SetFocus (hLName);
```

```
}
```

```
else
```

```
//year, name, SSN OK, send header query info to server
```

```
{
```

```
username = USER;
```

```
/** send the header query info to a file and let the server know **/
```

```
qf = fopen(QFILE, "wb");
```

```

if (qf == NULL)          //if file not available drop out of all if-else
    BLD_QueryTxtDlgFunc(hDlg,message,wParam,lParam);
else                      //if file available continue
{
    strcpy(hquery.Year, QYear);
    strcpy(hquery.EIN, QEIN);
    strcpy(hquery.Estab, QEstab);
    strcpy(hquery.seq_no, DEF_SEQ_NO);
    hquery.FName[0] = 0;
    hquery.LName[0] = 0;
    hquery.SSN[0] = 0;
    hquery.offset[0] = 0;
    fwrite(&hquery, sizeof(struct query), 1, qf);
    fclose(qf);

    /** clnt_create is one step down from a straight rpc call
        it is required here for control of the time out value **/
    clnt = clnt_create(HOST, GET_SEQ_HEADER, VERSNUM, UDP);
    if(clnt == NULL)
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Single Query, Get Seq Header, clnt_create");

            fprintf(ef, "%s\n", "CLIENT HANDLE IS NULL");
            fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
            fprintf(ef, "%d = T_ERRNO\n", t_errno);
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
            hYear = GetDlgItem(hDlg,IDMF_QYear);
            SetFocus(hYear);
        } //end of ef != NULL
    } //end of if clnt = NULL
    else /** set re-try timeout value for employer header request**

    {
        timeout.tv_sec = RETRY_TIME;
        timeout.tv_usec = 0;
        cres = clnt_control(clnt,CLSET_RETRY_TIMEOUT,(char _TKFAR *)&timeout

    );

    if (cres == 0)
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Single Query, Get Seq Header, clnt_contro

1");

            fprintf(ef, "%s\n\n", "RE-TRY TIMEOUT WAS NOT SET");
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
            hYear = GetDlgItem(hDlg,IDMF_QYear);
            SetFocus(hYear);
        } //end of ef != NULL
    } //end of if cres = 0
    else /** set total timeout & request employer header info **/

```



```

{
    timeout.tv_sec = TOT_TIME;
    timeout.tv_usec = 0;
    hrpcres = clnt_call(clnt, PROCNUM, (xdrproc_t)xdr_int,
        (caddr_t)&usernum, (xdrproc_t)xdr_int, (caddr_t)&hres, timeout
);
    clnt_destroy(clnt);
    if ((hrpcres != 0) || (hres != 1))        /** check for errors **/
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL)
            BLD_ErrorFileDlgFunc(hDlg, message, wParam, lParam);
        else
        {
            fprintf(ef, "\n%s\n", "Single Query, Get Seq Header, clnt_call
");
            fprintf(ef, "HRPCRES = %d\n", hrpcres);
            fprintf(ef, "HRES = %d\n", hres);
            if (hrpcres != 0)    /** if rpc call failed **/
            {
                /** rpc_createerr is a global variable returned by rpc_call
                    and relates the status of the call itself **/
                /** t_errno further delineates the error in certain settings
**/

                fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
                fprintf(ef, "%d = T_ERRNO\n\n", t_errno);
                BLD_QueryErrDlgFunc(hDlg, message, wParam, lParam);
                ShowCursor (FALSE);        //hide hourglass
                SetCursor (hCur);        //reset to arrow
                hYear = GetDlgItem(hDlg, IDMF_QYear);
                SetFocus (hYear);
            }
            else                /** if EIN could not be found **/
            {
                fprintf(ef, "%s\n\n", "SEARCH ENGINE COULD NOT FIND EIN");
                BLD_EINErrDlgFunc(hDlg, message, wParam, lParam);
                ShowCursor (FALSE);        //hide hourglass
                SetCursor (hCur);        //reset to arrow
                hEIN = GetDlgItem(hDlg, IDMF_QEIN);
                SetFocus (hEIN);
            }
            fclose(ef);
        } //end of ef != NULL
    } //end of error check (if errors, drop out of all if-else)
    else                //EIN verified, send single query info
    {
        /** send the single-query info to a file and let the server know
**/

        qf = fopen(QFILE, "wb");
        if (qf == NULL) //if file not available drop out of all if-else
            BLD_QueryTxtDlgFunc(hDlg, message, wParam, lParam);
        else                //if file available, continue
        {
            strcpy(squery.Year, QYear);
            strcpy(squery.EIN, QEIN);
            strcpy(squery.Estab, QEstab);
            squery.seq_no[0] = 0;
            strcpy(squery.FName, FName);
            strcpy(squery.LName, LName);
            strcpy(squery.SSN, QSSN);
            squery.offset[0] = 0;
            fwrite(&squery, sizeof(struct query), 1, qf);
            fclose(qf);

```

```

/** clnt_create is one step down from a straight rpc call
    it is required here for control of the time out value */
clnt = clnt_create(HOST, SINGLE_QUERY, VERSNUM, UDP);
if (clnt == NULL)
{
    ef = fopen(EFILE, "a");
    if (ef == NULL) //if error file not available alert user
        BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
    else //if error file available continue
    {
        fprintf(ef, "\n%s\n", "Single Query, Single Query, clnt_cr
eate");

        fprintf(ef, "%s\n", "CLIENT HANDLE IS NULL");
        fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
        fprintf(ef, "%d = T_ERRNO\n", t_errno);
        fclose(ef);
        BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
        ShowCursor (FALSE); //hide hourglass
        SetCursor (hCur); //reset to arrow
        hYear = GetDlgItem(hDlg,IDMF_QYear);
        SetFocus(hYear);
    } //end of ef != NULL
} //end of if clnt = NULL
else //** set re-try timeout value **/
{
    timeout.tv_sec = RETRY_TIME;
    timeout.tv_usec = 0;
    cres = clnt_control(clnt,CLSET_RETRY_TIMEOUT,(char _TKFAR *)
&timeout);

    if (cres == 0)
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Single Query, Single Query, clnt_c
ontrol");

            fprintf(ef, "%s\n\n", "RE-TRY TIMEOUT WAS NOT SET");
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
            hYear = GetDlgItem(hDlg,IDMF_QYear);
            SetFocus(hYear);
        } //end of ef != NULL
    } //end of if cres = 0
    else //** set total timeout & request search **/
    {
        timeout.tv_sec = TOT_TIME;
        timeout.tv_usec = 0;
        srpcres = clnt_call(clnt,PROCNUM,(xdrproc_t)xdr_int,
(caddr_t)&usernum,(xdrproc_t)xdr_int,(caddr_t)&sr
es, timeout);

        clnt_destroy(clnt);
        if ((srpcres != 0) || (sres != 1))
        {
            ef = fopen(EFILE, "a");
            if (ef == NULL) //if error file not available alert user
                BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
            else //if error file available continue
            {
                if (srpcres != 0)
                {

```

```

        fprintf(ef, "\n%s\n", "Single Query, Single Query, c
lnt_call");

        fprintf(ef, "%d = SRPCRES\n", srpcres );
        fprintf(ef, "%d = SRES\n\n", sres);
        fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
        fprintf(ef, "%d = T_ERRNO\n", t_errno);
        BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
        ShowCursor (FALSE);          //hide hourglass
        SetCursor (hCur); //reset to arrow
        hYear = GetDlgItem(hDlg,IDMF_QYear);
        SetFocus(hYear);
    }
else
{
    fprintf(ef, "\n\n%s\n", "Single Query, Single Query,
clnt_call");

    fprintf(ef, "NO MATCHES TO THIS QUERY\n\n");
    BLD_MatchErrDlgFunc(hDlg,message,wParam,lParam);
    ShowCursor (FALSE);          //hide hourglass
    SetCursor (hCur); //reset to arrow
    hYear = GetDlgItem(hDlg,IDMF_QYear);
    SetFocus(hYear);
}
fclose(ef);
} //end of ef != NULL
} //end of error check on srpcres and sres
else //parameters, communications & data verified
{
    //build next window if files available
    ShowCursor (FALSE);          //hide hourglass
    SetCursor (hCur);          //reset to arrow
    EndDialog(hDlg,IDOK);
    brfil = fopen(BRFILE, "rb"); //check file ptrs before
    hfil = fopen(HFILE, "rb");  //opening QINFO
    if ((hfil == NULL) || (brfil == NULL))
        BLD_MissingFileDlgFunc(hDlg,message,wParam,lParam);
    else //file available, build next window
    {
        fclose(hfil);
        fclose(brfil);
        BLD_OKDlgFunc(hDlg,message,wParam,lParam);
    }
} //everything executed properly
} //end of set total timeout value and request search
} //end of set re-try timeout value for single query
} //end of send single-query info if qf != NULL
} //end of EIN verified, send single-query info
} //end of set total timeout value for get employer header
} //end of set re-try timeout value for employer header request
} //end of send header query info if qf != NULL
} //end of send header query info
} //end of check name if entered
break; //end of IDOK

case IDCANCEL:
    fill_aggregate();
    write_aggregate(hDlg,message,wParam,lParam);
    if (!BLD_QUERYDlgDefault(hDlg,message,wParam,lParam))
        EndDialog(hDlg,IDCANCEL);
    break;

default:
    return BLD_QUERYDlgDefault(hDlg,message,wParam,lParam);
    break;
}
break;

```



```

        default:
            return BLD_QUERYDlgDefault(hDlg,message,wParam,lParam);
            break;
    }
    return TRUE; /* Did process the message */
} //END OF SINGLE QUERY ENTRY SCREEN

/*****
****/
/* displays matches to user single query
*/
/*****
****/

int BLD_OKDlgFunc(HWND,message,wParam,lParam) /* Startup procedure for modal dialog box */
HWND hWnd;
UINT message;
WPARAM wParam;
LONG lParam;
{
    FARPROC lpProc;
    int ReturnValue;

    lpProc = MakeProcInstance(BLD_OKDlgProc,hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)"QINFO", MainhWnd, lpProc);
    FreeProcInstance(lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,"QINFO",
            MB_OK | MB_ICONHAND);
    return ReturnValue;
}

BOOL FAR PASCAL BLD_OKDlgProc(hDlg, message, wParam, lParam) /* Modal dialog box procedure
*/
HWND hDlg;
UINT message;
WPARAM wParam;
LONG lParam;
{
    char BLFILE[16], //blanket file
        BRFILE[14], //browse file
        DFILE[14], //detail file
        HFILE[14], //header file
        QFILE[13]; //query file

    char c_usernum[3]; //user number converted to
                        //character for concatenation
                        //in file name

    char ListBoxString[200]; //** list box string **/
    char recoffset[30]; //** converted record offset **/

    CLIENT_TKFAR *clnt; //** handle to client object **/

    FILE *dfil, *brfil, *hfil; //** pointer to a file **/
    FILE *qf, *ef; //** pointer to query file **/

    HCURSOR hCur; //** cursor handle **/

    HWND hBl; //** handle to IDBlanket pushbutton **/
    HWND hLB; //** handle to list box **/

```

```

int arpcres, ares;          /** add_matches rpc call results **/
int count;                 /** counter **/
int drpcres, dres;         /** detail rpc call results **/
int hrpcres, hres;         /** header change rpc call results **/
int index;                 /** list box index of list box string **/
int tabs[] = {52,72,84,123,162,197,243,297,427}; /** tab settings for list box **/
int usernum, brpcres, bres, cres; /** user number, rpc call result, and head
er                           call result for blanket**/

long int lbitems;          /** # of items in list box **/
long num;                  /** holds results of fread or fseek **/

struct query aquery;       /** structure to hold add_matches query
                             parameters **/
struct query bquery;       /** structure to hold header query params
**/
struct query dquery;       /** structure to hold detail query params
**/
struct query hquery;       /** structure to hold header query params
**/
struct timeval timeout;    /** holds time variables for client calls
**/
struct W2Browse *pBrowse;  /** pointer to a browse structure **/

unsigned long offset;      /** location of record w/i a file **/

WORD numtabs = 9;         /** number of tabs in list box **/

//build file names using current user number
strcpy(BLFILE, "d:blank");
strcat(BLFILE, itoa(USER, c_usernum, 10) );
strcat(BLFILE, ".txt\0");

strcpy(BRFILE, "d:brow");
strcat(BRFILE, itoa(USER, c_usernum, 10) );
strcat(BRFILE, ".txt\0");

strcpy(DFILE, "d:det1");
strcat(DFILE, itoa(USER, c_usernum, 10) );
strcat(DFILE, ".txt\0");

strcpy(HFILE, "d:hdr");
strcat(HFILE, itoa(USER, c_usernum, 10) );
strcat(HFILE, ".txt\0");

strcpy(QFILE, "d:query");
strcat(QFILE, itoa(USER, c_usernum, 10) );
strcat(QFILE, ".txt\0");

switch(message)
{
case WM_INITDIALOG:
    //store start single query "qinfo" time
    current.start_qinfo = time(NULL);

    hfil = fopen(HFILE, "rb");
    if (hfil == NULL) //if file pointer null alert user
    {
        BLD_NULLPtrDlgFunc(hDlg,message,wParam,lParam);
        EndDialog(hDlg,1);
    }

```

```
    return TRUE;
}
else //if file available continue
{
    fread(&CurrEmprInfo, sizeof(struct W2EmprInfo), 1, hfil);
    fclose(hfil);
    CreateHeaderString();
    SetDlgItemText(hDlg, ID_QEmprHeader, HeaderString);
    SendDlgItemMessage(hDlg, IDLB_QMatch, LB_SETHORIZONTALEXTENT, 1000, 0);
    SendDlgItemMessage(hDlg, IDLB_QMatch, LB_SETTABSTOPS, numtabs,
        (LONG) (LPINT) tabs);

    offcount = 0; //initialize counter for entries in list box
    brfil = fopen(BRFILE, "rb");
    pBrowse = &Browse;
    if (brfil == NULL) //if file pointer null alert user
    {
        BLD_NULLPtrDlgFunc(hDlg, message, wParam, lParam);
        EndDialog(hDlg, 1);
        return TRUE;
    }
    else //if file available continue
    {
        for (count=1; count <= MAXQLB; count++) //** read from the file **/
        {
            offcount++; //set counter
            num = fread(pBrowse, sizeof(struct W2Browse), 1, brfil);

            if(num != 0) //if file contains data read
            {
                offset = pBrowse->record_loc[0]; //** convert the **/
                offset = offset<< 8; //** record location **/
                offset = offset + pBrowse->record_loc[1]; //** because of **/
                offset = offset<<8; //** different byte **/
                offset = offset + pBrowse->record_loc[2]; //** ordering on **/
                offset = offset<<8; //** the SUN **/
                offset = offset + pBrowse->record_loc[3];

                //** display the browse info in the list box **/
                ListBoxString[0] = 0;
                sprintf(ListBoxString, "%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s",
                    Browse.MRN,
                    Browse.seq_no,
                    Browse.wage_type,
                    Browse.AnnFICAWages,
                    Browse.AnnFICATips,
                    Browse.FICATaxWheld,
                    Browse.AnnWgsTpsOther,
                    Browse.EmpSSN,
                    Browse.EmpName,
                    ultoa(offset, reoffset, 10));
                SendDlgItemMessage(hDlg, IDLB_QMatch, LB_ADDSTRING, 0,
                    (LONG) (LPSTR) ListBoxString);
            } //end of if num != 0
        } //end of for loop - reading from the file

        fclose(brfil);
        SendDlgItemMessage(hDlg, IDLB_QMatch, LB_SETCURSEL, (WPARAM) 0, (LONG) 0);
        return BLD_OKDlgDefault(hDlg, message, wParam, lParam);
    } //end of if brfil != NULL
} // end of if hfil != NULL
break; //end of WM_INITDIALOG
```



```

case WM_COMMAND:
    switch(wParam)
    {

case ID_Change:        //when user presses arrow icon to change employer header
    aggregate.tot_diff_report++; //increment # of times diff. report selected
    strcpy(sequence, CurrEmprInfo.seq_no); //store current sequence number
                                //for display in data entry field

    BLD_SequenceDlgFunc(hDlg,message,wParam,lParam); //get desired seq_

no
    if ( (strcmp(sequence, CurrEmprInfo.seq_no)) != 0 ) //if not the same
get &
    { //display new info
    usernum = USER;
    hCur = SetCursor (LoadCursor (NULL, IDC_WAIT)); //change to hourgl
ass
    ShowCursor (TRUE); //show hourglass

    /** send the header query info to a file and let the server know **/
    qf = fopen(QFILE, "wb");
    if (qf == NULL) //if file not available drop out of all if-else
        BLD_QueryTxtDlgFunc(hDlg,message,wParam,lParam);
    else //if file available continue
    {
        strcpy(hquery.Year, CurrEmprInfo.ReportYear);
        strcpy(hquery.EIN, CurrEmprInfo.EIN);
        strcpy(hquery.Estab, CurrEmprInfo.EstabNumber);
        strcpy(hquery.seq_no, sequence);
        hquery.FName[0] = 0;
        hquery.LName[0] = 0;
        hquery.SSN[0] = 0;
        hquery.offset[0] = 0;
        fwrite(&hquery, sizeof(struct query), 1, qf);
        fclose(qf);

        /** clnt_create is one step down from a straight rpc call
            it is required here for control of the time out value **/
        clnt = clnt_create(HOST, GET_SEQ_HEADER, VERSNUM, UDP);
        if(clnt == NULL)
        {
            ef = fopen(EFILE, "a");
            if (ef == NULL) //if error file not available alert user
                BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
            else //if error file available continue
            {
                fprintf(ef, "\n%s\n", "Change Header, Get Seq Header, clnt_create"

);
                fprintf(ef, "%s\n", "CLIENT HANDLE IS NULL");
                fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
                fprintf(ef, "%d = T_ERRNO\n", t_errno);
                fclose(ef);
                BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
                ShowCursor (FALSE); //hide hourglass
                SetCursor (hCur); //reset to arrow
                hLB=GetDlgItem(hDlg, IDLB_QMatch);
                SetFocus(hLB);
            } //end of ef != NULL
        } //end of if clnt = NULL
    else //** set re-try timeout value for employer header request**
    {
        timeout.tv_sec = RETRY_TIME;
        timeout.tv_usec = 0;

```

```

    cres = clnt_control(clnt, CLSET_RETRY_TIMEOUT, (char _TKFAR *)&timeout
);
    if (cres == 0)
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg, message, wParam, lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Change Header, Get Seq Header, clnt_contr
ol");

            fprintf(ef, "%s\n\n", "RE-TRY TIMEOUT WAS NOT SET");
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg, message, wParam, lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
            hLB=GetDlgItem(hDlg, IDLB_QMatch);
            SetFocus(hLB);
        } //end of ef != NULL
    } //end of if cres = 0
    else /** set total timeout & request employer header info */
    {
        timeout.tv_sec = TOT_TIME;
        timeout.tv_usec = 0;
        hrpcres = clnt_call(clnt, PROCNUM, (xdrproc_t)xdr_int,
            (caddr_t)&usernum, (xdrproc_t)xdr_int, (caddr_t)&hres, timeout
);

        clnt_destroy(clnt);
        if ((hrpcres != 0) || (hres != 1)) /** check for errors */
        {
            ef = fopen(EFILE, "a");
            if (ef == NULL)
                BLD_ErrorFileDlgFunc(hDlg, message, wParam, lParam);
            else
            {
                fprintf(ef, "\n%s\n", "Change Header, Get Seq Header, clnt_cal
1");

                fprintf(ef, "HRPCRES = %d\n", hrpcres );
                fprintf(ef, "HRES = %d\n", hres);
                if (hrpcres != 0) /** if rpc call failed */
                {
                    /** rpc_createerr is a global variable returned by rpc_call
                    and relates the status of the call itself */
                    /** t_errno further delineates the error in certain settings
                    */

                    fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
                    fprintf(ef, "%d = T_ERRNO\n\n", t_errno);
                    BLD_QueryErrDlgFunc(hDlg, message, wParam, lParam);
                    ShowCursor (FALSE); //hide hourglass
                    SetCursor (hCur); //reset to arrow
                    hLB=GetDlgItem(hDlg, IDLB_QMatch);
                    SetFocus(hLB);
                }
            }
            else /** if sequence no. could not be found */
            {
                fprintf(ef, "%s\n\n", "SEARCH ENGINE COULD NOT FIND SEQUENCE
NUMBER");

                BLD_SeqErrDlgFunc(hDlg, message, wParam, lParam);
                ShowCursor (FALSE); //hide hourglass
                SetCursor (hCur); //reset to arrow
                hLB=GetDlgItem(hDlg, IDLB_QMatch);
                SetFocus(hLB);
            }
        }
        fclose(ef);
    }

```

```

    } //end of ef != NULL
  } //end of error check (if errors, drop out of all if-else)
else
{
  hfil = fopen(HFILE, "rb");
  if (hfil == NULL) //if file pointer null alert user
  {
    BLD_HFileDlgFunc(hDlg,message,wParam,lParam);
    ShowCursor (FALSE);           //hide hourglass
    SetCursor (hCur);            //reset to arrow
    hLB=GetDlgItem(hDlg, IDLB_QMatch);
    SetFocus(hLB);
  }
  else //if file available continue
  {
    fread(&CurrEmprInfo, sizeof(struct W2EmprInfo), 1, hfil);
    fclose(hfil);
    CreateHeaderString();
    SetDlgItemText(hDlg,ID_QEmprHeader,HeaderString);
    ShowCursor (FALSE);           //hide hourglass
    SetCursor (hCur);            //reset to arrow
    hLB=GetDlgItem(hDlg, IDLB_QMatch);
    SetFocus(hLB);
  }
  } //everything executed property, new employer header displayed
} //end of set total timeout and request employer header
} //end of set retry timeout for employer header
} //end of if qf != NULL
} //end of if sequence number not the same
else
{
  hLB=GetDlgItem(hDlg, IDLB_QMatch);
  SetFocus(hLB);
}
break; //end of ID_Change

```

```

case IDOK: /** when user hits return in listbox */
  hCur = SetCursor (LoadCursor (NULL, IDC_WAIT)); //change to hourglass
  ShowCursor (TRUE);                               //show hourglass

  ListBoxString[0] = 0;                            /** use index to get string then

  index = (WORD) SendDlgItemMessage(hDlg,           /** parse out the offset */
    IDLB_QMatch, LB_GETCURSEL, 0, 0);              /** and the sequence number */
  SendDlgItemMessage(hDlg, IDLB_QMatch, LB_GETTEXT,
    index, (LONG) (LPSTR) ListBoxString);
  strncpy(recoffset, ListBoxString + 95, 29);
  recoffset[29] = 0;                                /** terminate string */
  strncpy(dquery.seq_no, ListBoxString + 12, 3);
  dquery.seq_no[3] = 0;                             /** terminate string */

  usernum = USER;
  /** send the detail query info to a file and let the server know */
  qf = fopen(QFILE, "wb");
  if (qf == NULL) //if file not available drop out of all if-else
    BLD_QueryTxtDlgFunc(hDlg,message,wParam,lParam);
  else //if file available continue
  {
    strcpy(dquery.Year, QYear);
    strcpy(dquery.EIN, QEIN);
    strcpy(dquery.Estab, QEstab);
    dquery.FName[0] = 0;
  }

```



```
dquery.LName[0] = 0;
dquery.SSN[0] = 0;
strcpy(dquery.offset, reoffset);
fwrite(&dquery, sizeof(struct query), 1, qf);
fclose(qf);

/** clnt_create is one step down from a straight rpc call
    it is required here for control of the time out value **/
clnt = clnt_create(HOST, GET_EMPL_DETAIL, VERSNUM, UDP);
if (clnt == NULL)
{
    ef = fopen(EFILE, "a");
    if (ef == NULL) //if error file not available alert user
        BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
    else //if error file available continue
    {
        fprintf(ef, "\n%s\n", "Query Info, Get_Empl_Detail, clnt_create");
        fprintf(ef, "%s\n", "CLIENT HANDLE IS NULL");
        fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
        fprintf(ef, "%d = T_ERRNO\n", t_errno);
        fclose(ef);
        BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
        ShowCursor (FALSE); //hide hourglass
        SetCursor (hCur); //reset to arrow
        hLB=GetDlgItem(hDlg, IDLB_QMatch);
        SetFocus(hLB);
    } //end of ef != NULL
} //end of if clnt = NULL
else //** set re-try timeout value **/
{
    timeout.tv_sec = RETRY_TIME;
    timeout.tv_usec = 0;
    cres = clnt_control(clnt,CLSET_RETRY_TIMEOUT,(char _TKFAR *)&timeout);
    if (cres == 0)
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Query Info, Get_Empl_Detail, clnt_control");

            fprintf(ef, "%s\n\n", "RE-TRY TIMEOUT WAS NOT SET");
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
            hLB=GetDlgItem(hDlg, IDLB_QMatch);
            SetFocus(hLB);
        } //end of ef != NULL
    } //end of if cres = 0
else //** set total timeout & request employee detail **/
{
    timeout.tv_sec = TOT_TIME;
    timeout.tv_usec = 0;
    drpcres = clnt_call(clnt,PROCNUM,(xdrproc_t)xdr_int,
                        (caddr_t)&usernum, (xdrproc_t)xdr_int,
                        (caddr_t)&dres, timeout);
    clnt_destroy(clnt);
    if ((drpcres != 0) || (dres != 1))
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
    }
}
```

```

else    //if error file available continue
{
    fprintf(ef, "\n%s\n", "Single, Get_Empl_Detail, clnt_call");
    fprintf(ef, "%d = DRPCRES\n", drpcres );
    fprintf(ef, "%d = DRES\n\n", dres);
    fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
    fprintf(ef, "%d = T_ERRNO\n", t_errno);
    fclose(ef);
    BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
    ShowCursor (FALSE);    //hide hourglass
    SetCursor (hCur);    //reset to arrow
    hLB=GetDlgItem(hDlg, IDLB_QMatch);
    SetFocus(hLB);
} //end of ef != NULL
} //end of error check on drpcres and dres
else //parameters, communications & data verified
{
    dfil = fopen(DFILE, "rb");    /** open the detail file **/
    if (dfil == NULL)    //check file ptr
    {
        BLD_DFileErrDlgFunc(hDlg,message,wParam,lParam);
        ShowCursor(FALSE);    //hide hourglass
        SetCursor (hCur);    //reset to arrow
        hLB=GetDlgItem(hDlg, IDLB_QMatch);
        SetFocus(hLB);
    }
    else
    {
        /** read in employee detail info **/
        fread(&EDetail, sizeof(struct W2EmpInfo), 1, dfil);
        fclose(dfil);
        CreateEDetail();
        BLD_EmployeeDetailDlgFunc(hDlg,message,wParam,lParam);
        ShowCursor(FALSE);    //hide hourglass
        SetCursor (hCur);    //reset to arrow
        hLB=GetDlgItem(hDlg, IDLB_QMatch);
        SetFocus(hLB);
    }
} //everything executed properly
} //end of set total timeout
} //end of set re-try timeout value
} //end of qf != NULL
break; //end of IDOK

case IDCANCEL:
    if (!BLD_OKDlgDefault(hDlg,message,wParam,lParam))
        EndDialog(hDlg, IDCANCEL);
    break; //end of IDCANCEL

case ID_CLOSE: //when user presses the close button
    current.stop_single = time(NULL); //store single query stop time
    BLD_questDlgFunc(hDlg,message,wParam,lParam);
    EndDialog(hDlg, ID_CLOSE);
    break; //end of ID_CLOSE

case IDBlanket: //when user presses potential blanket button
    current.stop_single = time(NULL); //store single query stop time
    aggregate.tot_blanket++; //increment # of times potential blanket selected
    BLD_questDlgFunc(hDlg,message,wParam,lParam);
    usernum = USER;
    strcpy(sequence, CurrEmprInfo.seq_no); //get current seq_no
    BLD_SequenceDlgFunc(hDlg,message,wParam,lParam); //get new seq_no

    /** send the header query info to a file and let the server know **/
    qf = fopen(QFILE, "wb");

```

```

if (qf == NULL) //if file not available drop out of all if-else
    BLD_QueryTxtDlgFunc(hDlg,message,wParam,lParam);
else //if file available continue
{
    strcpy(bquery.Year, CurrEmprInfo.ReportYear);
    strcpy(bquery.EIN, CurrEmprInfo.EIN);
    strcpy(bquery.Estab, CurrEmprInfo.EstabNumber);
    strcpy(bquery.seq_no, sequence);
    bquery.FName[0] = 0;
    bquery.LName[0] = 0;
    bquery.SSN[0] = 0;
    fwrite(&bquery, sizeof(struct query), 1, qf);
    fclose(qf);
    hCur = SetCursor (LoadCursor (NULL, IDC_WAIT)); //change to hourglass
    ShowCursor (TRUE); //show hourglass
    clnt = clnt_create(HOST, GET_BLANKET, VERSNUM, UDP);
    if(clnt == NULL)
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Query Info, Get Blanket, clnt_create");
            fprintf(ef, "%s\n\n", "CLIENT HANDLE IS NULL");
            fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
            fprintf(ef, "%d = T_ERRNO\n\n", t_errno);
            fclose(ef);
            BLD_BlanketErrDlgFunc(hDlg,message,wParam,lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
            hBl = GetDlgItem(hDlg,IDBlanket);
            SetFocus(hBl);
        } //end of if ef != NULL
    } //end of if clnt = NULL
else
{
    timeout.tv_sec = RETRY_TIME; /** set re-try timeout value **/
    timeout.tv_usec = 0;
    cres = clnt_control(clnt, CLSET_RETRY_TIMEOUT, (char_TKFAR *)&timeout);
    if (cres == 0) //check for errors
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Query Info, Get Blanket, clnt_control");
            fprintf(ef, "%s\n\n", "RE-TRY TIMEOUT WAS NOT SET");
            fclose(ef);
            BLD_BlanketErrDlgFunc(hDlg,message,wParam,lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
            hBl = GetDlgItem(hDlg,IDBlanket);
            SetFocus(hBl);
        } //end of if ef != NULL
    } //end of if cres = 0
else
{
    timeout.tv_sec = TOT_TIME; /** set total timeout & request blanket in
fo **/
    timeout.tv_usec = 0;

    brpcres = clnt_call(clnt,PROCNUM,(xdrproc_t)xdr_int,

```



```

        (caddr_t)&usernum, (xdrproc_t)xdr_int, (caddr_t)&bres, timeout);
    clnt_destroy(clnt);

    if ((brpcres != 0) || (bres != 1))           //check for errors
        /** rpc_createrr is a global variable returned by rpc_call
            and relates the status of the call itself **/
        /** t_errno further delineates the error in certain settings **/
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Query Info, Get Blanket, clnt_call");
            fprintf(ef, "%d = BRPCRES\n", brpcres);
            fprintf(ef, "%d = BRES\n", bres);
            if (brpcres != 0) //if rpc call failed
            {
                fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
                fprintf(ef, "%d = T_ERRNO\n\n", t_errno);
                BLD_BlanketErrDlgFunc(hDlg,message,wParam,lParam);
                ShowCursor (FALSE);           //hide hourglass
                SetCursor (hCur);           //reset to arrow
                hBl = GetDlgItem(hDlg,IDBlanket);
                SetFocus(hBl);
            }
            else //if sequence no. could not be found
            {
                fprintf(ef, "%s\n\n", "SEARCH ENGINE COULD NOT FIND SEQUENCE NUM
BER");
                BLD_SeqErrDlgFunc(hDlg,message,wParam,lParam);
                ShowCursor (FALSE); //hide hourglass
                SetCursor (hCur); //reset to arrow
                hLB=GetDlgItem(hDlg, IDLB_QMatch);
                SetFocus(hLB);
            }
            fclose(ef);
        } //end of ef != NULL
    } //end of if brpcres or bres incorrect
else
{
    ShowCursor (FALSE);           //hide hourglass
    SetCursor (hCur);           //reset to arrow
    EndDialog(hDlg,1);
    dfil = fopen(BLFILE, "rb"); //check file ptr before opening BI
    if (dfil == NULL)
        BLD_BlankErrDlgFunc(hDlg,message,wParam,lParam);
    else
    {
        fclose(dfil);
        BLD_Function6DlgFunc(hDlg,message,wParam,lParam);
    }
} //end of verification
} //end of set total timeout value
} //end of set re-try timeout value
} //end of if header query file available
break; //end of IDBlanket

case IDLB_QMatch: //** when user double-clicks in listbox **/
if (HIWORD(lParam)==LBN_DBLCLK)
{
    hCur = SetCursor (LoadCursor (NULL, IDC_WAIT)); //change to hourglass
    ShowCursor (TRUE); //show hourglass

```

```
ListBoxString[0] = 0;                                /** use index to get string then
**/
index = (WORD) SendDlgItemMessage(hDlg,              /** parse out the offset **/
    IDLB_QMatch, LB_GETCURSEL, 0, 0);                /** and the sequence number **/
SendDlgItemMessage(hDlg, IDLB_QMatch, LB_GETTEXT,
    index, (LONG) (LPSTR) ListBoxString);
strncpy(recoffset, ListBoxString + 95, 29);
recoffset[29] = 0;                                    /** terminate string **/
strncpy(dquery.seq_no, ListBoxString + 12, 3);
dquery.seq_no[3] = 0;                                /** terminate string **/

usernum = USER;
/** send the detail query info to a file and let the server know **/
qf = fopen(QFILE, "wb");
if (qf == NULL) //if file not available drop out of all if-else
    BLD_QueryTxtDlgFunc(hDlg, message, wParam, lParam);
else //if file available continue
{
    strcpy(dquery.Year, QYear);
    strcpy(dquery.EIN, QEIN);
    strcpy(dquery.Estab, QEstab);
    dquery.FName[0] = 0;
    dquery.LName[0] = 0;
    dquery.SSN[0] = 0;
    strcpy(dquery.offset, recoffset);
    fwrite(&dquery, sizeof(struct query), 1, qf);
    fclose(qf);

    /** clnt_create is one step down from a straight rpc call
        it is required here for control of the time out value **/
    clnt = clnt_create(HOST, GET_EMPL_DETAIL, VERSNUM, UDP);
    if (clnt == NULL)
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg, message, wParam, lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Query Info, Get_Empl_Detail, clnt_create");
            fprintf(ef, "%s\n", "CLIENT HANDLE IS NULL");
            fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
            fprintf(ef, "%d = T_ERRNO\n", t_errno);
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg, message, wParam, lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
            hLB=GetDlgItem(hDlg, IDLB_QMatch);
            SetFocus(hLB);
        } //end of ef != NULL
    } //end of if clnt = NULL
    else /** set re-try timeout value **/
    {
        timeout.tv_sec = RETRY_TIME;
        timeout.tv_usec = 0;
        cres = clnt_control(clnt, CLSET_RETRY_TIMEOUT, (char _TKFAR *)&timeout);
        if (cres == 0)
        {
            ef = fopen(EFILE, "a");
            if (ef == NULL) //if error file not available alert user
                BLD_ErrorFileDlgFunc(hDlg, message, wParam, lParam);
            else //if error file available continue
            {
                fprintf(ef, "\n%s\n", "Query Info, Get_Empl_Detail, clnt_control")
            }
        }
    }
}
```

```
;

    fprintf(ef, "%s\n\n", "RE-TRY TIMEOUT WAS NOT SET");
    fclose(ef);
    BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
    ShowCursor (FALSE);      //hide hourglass
    SetCursor (hCur);       //reset to arrow
    hLB=GetDlgItem(hDlg, IDLB_QMatch);
    SetFocus(hLB);
} //end of ef != NULL
} //end of if cres = 0
else      /** set total timeout & request employe detail info**/
{
    timeout.tv_sec = TOT_TIME;
    timeout.tv_usec = 0;
    drpcres = clnt_call(clnt,PROCNUM,(xdrproc_t)xdr_int,
                      (caddr_t)&usernum, (xdrproc_t)xdr_int,
                      (caddr_t)&dres, timeout);
    clnt_destroy(clnt);
    if ((drpcres != 0) || (dres != 1))
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Query Info, Get_Empl_Detail, clnt_call");
            fprintf(ef, "%d = DRPCRES\n", drpcres);
            fprintf(ef, "%d = DRES\n\n", dres);
            fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
            fprintf(ef, "%d = T_ERRNO\n\n", t_errno);
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
            ShowCursor (FALSE);      //hide hourglass
            SetCursor (hCur);       //reset to arrow
            hLB=GetDlgItem(hDlg, IDLB_QMatch);
            SetFocus(hLB);
        } //end of ef != NULL
    } //end of error check on drpcres and dres
else //parameters, communications & data verified
{
    dfil = fopen(DFILE, "rb");      /** open the detail file **/
    if (dfil == NULL)              //check file ptr
    {
        BLD_DFileErrDlgFunc(hDlg,message,wParam,lParam);
        ShowCursor(FALSE);          //hide hourglass
        SetCursor (hCur);          //reset to arrow
        hLB=GetDlgItem(hDlg,IDLB_QMatch);
        SetFocus(hLB);
    }
else
    {
        /** read in employe info **/
        fread(&EDetail, sizeof(struct W2EmpInfo), 1, dfil);
        fclose(dfil);
        CreateEDetail();
        BLD_EmployeeDetailDlgFunc(hDlg,message,wParam,lParam);
        ShowCursor(FALSE);          //hide hourglass
        SetCursor (hCur);          //reset to arrow
        hLB=GetDlgItem(hDlg, IDLB_QMatch);
        SetFocus(hLB);
    }
} //everything executed properly
} //end of set total timeout
} //end of set re-try timeout value
} //end of qf != NULL
```



```
    } //end of if double-clicked
    break; //end of IDLB_QMatch
```

```
case ID_QAddMatch:    /** when user presses additional matches button **/
                    //file availability previously checked
```

```
    current.add_match++; //increment add matches counter
```

```
    //get # of items in listbox
    lbitems = SendDlgItemMessage(hDlg, IDLB_QMatch, LB_GETCOUNT, 0, 0);
    ltoa(lbitems, reoffset, 10); //convert to string for storage in aquery
```

```
    usernum = USER;
    /** send the add_matches query info to a file and let the server know **/
    qf = fopen(QFILE, "wb");
    if (qf == NULL) //if file not available drop out of all if-else
        BLD_QueryTxtDlgFunc(hDlg, message, wParam, lParam);
    else //if file available continue
```

```
    {
        strcpy(aquery.Year, QYear);
        strcpy(aquery.EIN, QEIN);
        strcpy(aquery.Estab, QEstab);
        aquery.FName[0] = 0;
        aquery.LName[0] = 0;
        aquery.SSN[0] = 0;
        aquery.seq_no[0] = 0;
        strcpy(aquery.offset, reoffset); //in this case, reoffset = lbitems
        fwrite(&aquery, sizeof(struct query), 1, qf);
        fclose(qf);
```

```
        hCur = SetCursor (LoadCursor (NULL, IDC_WAIT)); //change to hourgl
```

```
    ass
```

```
        ShowCursor (TRUE); //show hourglass
```

```
    /** clnt_create is one step down from a straight rpc call
        it is required here for control of the time out value **/
    clnt = clnt_create(HOST, ADD_MATCHES, VERSNUM, UDP);
    if(clnt == NULL)
```

```
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg, message, wParam, lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Query Info, Add_Matches, clnt_create");
            fprintf(ef, "%s\n", "CLIENT HANDLE IS NULL");
            fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
            fprintf(ef, "%d = T_ERRNO\n", t_errno);
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg, message, wParam, lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
            hLB=GetDlgItem(hDlg, IDLB_QMatch);
            SetFocus(hLB);
        } //end of ef != NULL
    } //end of if clnt = NULL
```

```
    else //** set re-try timeout value **/
    {
        timeout.tv_sec = RETRY_TIME;
        timeout.tv_usec = 0;
        cres = clnt_control(clnt, CLSET_RETRY_TIMEOUT, (char _TKFAR *)&timeout);
        if (cres == 0)
```

```
{
    ef = fopen(EFILE, "a");
    if (ef == NULL) //if error file not available alert user
        BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
    else //if error file available continue
    {
        fprintf(ef, "\n%s\n", "Query Info, Add_Matches, clnt_control");
        fprintf(ef, "%s\n\n", "RE-TRY TIMEOUT WAS NOT SET");
        fclose(ef);
        BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
        ShowCursor (FALSE); //hide hourglass
        SetCursor (hCur); //reset to arrow
        hLB=GetDlgItem(hDlg, IDLB_QMatch);
        SetFocus(hLB);
    } //end of ef != NULL
} //end of if cres = 0
else /** set total timeout & request add matches info**/
{
    timeout.tv_sec = TOT_TIME;
    timeout.tv_usec = 0;
    arpcres = clnt_call(clnt,PROCNUM,(xdrproc_t)xdr_int,
                        (caddr_t)&usernum,(xdrproc_t)xdr_int,
                        (caddr_t)&ares, timeout);

    clnt_destroy(clnt);
    if ((arpcres != 0) || (ares != 1))
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
        else //if error file available continue
        {
            if (arpcres != 0) //if rpc error
            {
                fprintf(ef, "\n%s\n", "Query Info, Add_Matches, clnt_call");
                fprintf(ef, "%d = ARPCRES\n", arpcres);
                fprintf(ef, "%d = ARES\n\n", ares);
                fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
                fprintf(ef, "%d = T_ERRNO\n\n", t_errno);
                fclose(ef);
                BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
                ShowCursor (FALSE); //hide hourglass
                SetCursor (hCur); //reset to arrow
                hLB=GetDlgItem(hDlg, IDLB_QMatch);
                SetFocus(hLB);
            }
            else //if result error
            {
                current.add_match--; //adjust count if user presses
                                    //add matches by error or if
                                    //no more matches exist and listbox
                                    //was not added to

                fprintf(ef, "\n%s\n", "Query Info, Add_Matches, clnt_call");
                fprintf(ef, "NO MORE MATCHES FOUND\n");
                fprintf(ef, "%d = ARES\n\n", ares);
                fclose(ef);
                BLD_NoMoreMatchesDlgFunc(hDlg,message,wParam,lParam);
                ShowCursor (FALSE); //hide hourglass
                SetCursor (hCur); //reset to arrow
                hLB=GetDlgItem(hDlg, IDLB_QMatch);
                SetFocus(hLB);
            }
        } //end of ef != NULL
    } //end of error check on arpcres and ares
}
```

```

else //parameters, communications & data verified, fill list box
{
    brfil = fopen(BRFILE, "rb");           /** open the browse file */
    /*
    if (brfil == NULL) //if file not available alert user
    {
        BLD_DataErrDlgFunc(hDlg,message,wParam,lParam);
        ShowCursor (FALSE);           //hide hourglass
        SetCursor (hCur);           //reset to arrow
        hLB=GetDlgItem(hDlg, IDLB_QMatch);
        SetFocus (hLB);
    }
    else //if file available continue
    {
        pBrowse = &Browse;
        for (count=1; count <=MAXQLB; count++) /** read from the file */
        {
            num = fread(pBrowse, sizeof(struct W2Browse), 1, brfil);
            if(num != 0) //if file contains data or not EOF
            {
                offset = pBrowse->record_loc[0];           /** convert the */
                offset = offset<< 8;                       /** record location */

                offset = offset + pBrowse->record_loc[1]; /** because of */
                offset = offset<<8;                       /** different byte */

                offset = offset + pBrowse->record_loc[2]; /** ordering on */
                offset = offset<<8;                       /** the SUN */
                offset = offset + pBrowse->record_loc[3];

                /** display the browse info in the list box */
                ListBoxString[0] = 0;
                sprintf(ListBoxString,"%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s",
                    Browse.MRN,
                    Browse.seq_no,
                    Browse.wage_type,
                    Browse.AnnFICAWages,
                    Browse.AnnFICATips,
                    Browse.FICATaxWheld,
                    Browse.AnnWgsTpsOther,
                    Browse.EmpSSN,
                    Browse.EmpName,
                    ultoa(offset, reoffset, 10));
                SendDlgItemMessage(hDlg,IDLB_QMatch,LB_ADDSTRING,0,
                    (LONG) (LPSTR)ListBoxString);
            } //end of if file contains data or not EOF
        } //end of for loop
        fclose(brfil);
        ShowCursor (FALSE);           //hide hourglass
        SetCursor (hCur);           //reset to arrow
        hLB = GetDlgItem(hDlg,IDLB_QMatch);
        SetFocus (hLB);
        SendDlgItemMessage(hDlg,IDLB_QMatch,LB_SETCURSEL,
            (WPARAM) lbitems, (LONG) 0);
    } //end of if brfil != NULL

    } //everything executed properly
} //end of set total timeout
} //end of set re-try timeout value
} //end of qf != NULL
break; //end of ID_QAddMatch

default:
    return BLD_OKDlgDefault(hDlg,message,wParam,lParam);

```



```

        break;
    }
    break;

default:
    return BLD_OKDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE; /* Did process the message */
} //END OF DISPLAY MATCHES TO SINGLE QUERY

/*****
/* displays records for blanket report */
*****/

int BLD_Function6DlgFunc(hWnd,message,wParam,lParam) /* Startup procedure for modal dialog
box */
HWND hWnd;
UINT message;
WPARAM wParam;
LONG lParam;
{
    FARPROC lpProc;
    int ReturnValue;

    lpProc = MakeProcInstance(BLD_Function6DlgProc,hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)"BINFO", MainhWnd, lpProc);
    FreeProcInstance(lpProc);
    if (ReturnValue==-1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,"BINFO",
            MB_OK | MB_ICONHAND);
    return ReturnValue;
}

BOOL FAR PASCAL BLD_Function6DlgProc(hDlg, message, wParam, lParam) /* Modal dialog box pr
cedure */
HWND hDlg;
UINT message;
WPARAM wParam;
LONG lParam;
{
    char BLFILE[16], //blanket file
        HFILE[14]; //header file

    char c_usernum[3]; //user number converted to
                    //character for concatenation
                    //in file name

    char ListBoxString[200], //** list box string **/
        reoffset[30]; //** converted record offset **/

    FILE *datafile, *hfil; //** pointers to data files **/

    HCURSOR hCur; //** cursor handle **/

    HWND hLB; //** handle to list box **/

    int count; //** counter **/
    int index; //** list box index of list box str
ing **/
    int tabs[] = {52,72,84,123,162,197,243,297,427}; //** tab settings for list box **/

```



```
        Blanket[count-1].AnnFICATips,
        Blanket[count-1].FICATaxWheld,
        Blanket[count-1].AnnWgsTpsOther,
        Blanket[count-1].EmpSSN,
        Blanket[count-1].EmpName,
        ultoa(offset, reoffset, 10));
    SendDlgItemMessage(hDlg, IDLB_BMatch, LB_ADDSTRING, 0,
        (LONG) (LPSTR) ListBoxString);
} //end of if num != 0
} //end of if not EOF

fclose(datafile);
SendDlgItemMessage(hDlg, IDLB_BMatch, LB_SETCURSEL, (WPARAM) 0, (LONG) 0);
return BLD_Function6DlgDefault(hDlg, message, wParam, lParam);
} //end of if datafil != NULL
} // end of if hfil != NULL
break; //end of WM_INITDIALOG

case WM_COMMAND:
    switch(wParam)
    {

case IDOK:                                /** when user hits return inside the listbox **/
    hCur = SetCursor (LoadCursor (NULL, IDC_WAIT)); //change to hourglass
    ShowCursor (TRUE); //show hourglass

    ListBoxString[0] = 0;
    index = (WORD) SendDlgItemMessage(hDlg,
        IDLB_BMatch, LB_GETCURSEL, 0, 0);
    strcpy(sequence, CurrEmprInfo.seq_no);

    CopyBlanket_EDetail(index);
    CreateEDetail();
    BLD_EmployeeDetailDlgFunc(hDlg, message, wParam, lParam);
    ShowCursor(FALSE); //hide hourglass
    SetCursor (hCur); //reset to arrow
    hLB=GetDlgItem(hDlg, IDLB_BMatch);
    SetFocus(hLB); //set focus back to listbox
    break; //end of IDOK

case IDCANCEL: //if user presses Esc
    if (!BLD_Function6DlgDefault(hDlg, message, wParam, lParam))
        EndDialog(hDlg, IDCANCEL);
    break;

case IDLB_BMatch: /** when user double clicks inside the listbox **/
    if (HIWORD(lParam)==LBN_DBLCLK)
    {
        hCur = SetCursor (LoadCursor (NULL, IDC_WAIT)); //change to hourglass
        ShowCursor (TRUE); //show hourglass

        ListBoxString[0] = 0;
        index = (WORD) SendDlgItemMessage(hDlg,
            IDLB_BMatch, LB_GETCURSEL, 0, 0);
        strcpy(sequence, CurrEmprInfo.seq_no);

        CopyBlanket_EDetail(index);
        CreateEDetail();
        BLD_EmployeeDetailDlgFunc(hDlg, message, wParam, lParam);
        ShowCursor(FALSE); //hide hourglass
        SetCursor (hCur); //reset to arrow
        hLB=GetDlgItem(hDlg, IDLB_BMatch);
```



```

        SetFocus(hLB);                                //set focus back to listbo
x
    }
    break; //end of IDLB_BMatch

    case IDPRINTB: //when user presses Print Blanket
        aggregate.tot_pr_blanket++; //increment # of times print blanket selected
        BLD_PrintBlanketDlgFunc(hDlg,message,wParam,lParam);
        break;

    case ID_CLOSE:
        write_aggregate(hDlg,message,wParam,lParam); //write aggregate stats to file
        EndDialog(hDlg, ID_CLOSE);
        break;

    default:
        return BLD_Function6DlgDefault(hDlg,message,wParam,lParam);
        break;
    }
    break;

default:
    return BLD_Function6DlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE; /* Did process the message */
} //END OF DISPLAY BLANKET RECORDS

```

```

/*****
/* prompts user to enter report info */
*****/

```

```

int BLD_ReportDlgFunc(hWnd,message,wParam,lParam) /* Startup procedure for modal dialog bo
x */
HWND hWnd;
UINT message;
WPARAM wParam;
LONG lParam;
{
    FARPROC lpProc;
    int ReturnValue;

    lpProc = MakeProcInstance((FARPROC)BLD_ReportDlgProc,hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)"REPORT",hWnd, lpProc);
    FreeProcInstance(lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,"REPORT",
            MB_OK | MB_ICONHAND);
    return ReturnValue;
}

```

```

BOOL FAR PASCAL BLD_ReportDlgProc(hDlg, message, wParam, lParam) /* Modal dialog box proce
dure */
HWND hDlg;
UINT message;
WPARAM wParam;
LONG lParam;
{
    switch(message)

```

```

{
case WM_INITDIALOG:
    return BLD_ReportDlgDefault (hDlg,message,wParam,lParam);
    break;

case WM_COMMAND:
    switch(wParam)
    {
    case IDOK:
        RYear[0] = 0;
        REIN[0] = 0;
        REstab[0] = 0;
        GetDlgItemText (hDlg, IDMF_RYear, (LPSTR)RYear,5);  /** get N+1 characters **
        /
        GetDlgItemText (hDlg, IDMF_REIN, (LPSTR)REIN,11);
        GetDlgItemText (hDlg, IDMF_REstab, (LPSTR)REstab,5);
        EndDialog (hDlg, IDOK);
        BLD_ReportStatisticsDlgFunc (hDlg,message,wParam,lParam);
        break;
    case IDCANCEL:
        if (!BLD_ReportDlgDefault (hDlg,message,wParam,lParam))
            EndDialog (hDlg, IDCANCEL);
        break;
    default:
        return BLD_ReportDlgDefault (hDlg,message,wParam,lParam);
        break;
    }
    break;

default:
    return BLD_ReportDlgDefault (hDlg,message,wParam,lParam);
    break;
}

return TRUE; /* Did process the message */
} //END OF REPORT ENTRY SCREEN

```

```

/*****
/
/* displays report statistics - not implemented in the prototype *
/
/*****
/

```

```

int BLD_ReportStatisticsDlgFunc (hWnd,message,wParam,lParam) /* Startup procedure for modal
    dialog box */
HWND hWnd;
UINT message;
WPARAM wParam;
LONG lParam;
{
    FARPROC lpProc;
    int ReturnValue;

    lpProc = MakeProcInstance ((FARPROC)BLD_ReportStatisticsDlgProc,hInst);
    ReturnValue = DialogBox (hInst, (LPSTR)"STATISTICS", MainhWnd, lpProc);
    FreeProcInstance (lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage (hWnd,BLD_CannotCreate,"STATISTICS",
            MB_OK | MB_ICONHAND);
    return ReturnValue;
}

```

```
BOOL FAR PASCAL BLD_ReportStatisticsDlgProc(hDlg, message, wParam, lParam) /* Modal dialog
box procedure */
HWND hDlg;
UINT message;
WPARAM wParam;
LONG lParam;
{
    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_ReportStatisticsDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            switch(wParam)
            {
                case IDOK:
                    if (!BLD_ReportStatisticsDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;
                case IDCANCEL:
                    if (!BLD_ReportStatisticsDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;
                default:
                    return BLD_ReportStatisticsDlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
            return BLD_ReportStatisticsDlgDefault(hDlg,message,wParam,lParam);
            break;
    }
    return TRUE; /* Did process the message */
} //END OF DISPLAY REPORT STATISTICS DIALOG
```

```
/* *****
/* prompts user to enter browse report info */
/* *****
```

```
int BLD_BrowseEntryDlgFunc(hWnd,message,wParam,lParam) /* Startup procedure for modal dial
og box */
HWND hWnd;
UINT message;
WPARAM wParam;
LONG lParam;
{
    FARPROC lpProc;
    int ReturnValue;

    lpProc = MakeProcInstance((FARPROC)BLD_BrowseEntryDlgProc,hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)"NEWBROWSE", hWnd, lpProc);
    FreeProcInstance(lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,"NEWBROWSE",
            MB_OK | MB_ICONHAND);
    return ReturnValue;
}
```

```
BOOL FAR PASCAL BLD_BrowseEntryDlgProc(hDlg, message, wParam, lParam) /* Modal dialog box
```



```
procedure */
HWND hDlg;
UINT message;
WPARAM wParam;
LONG lParam;
{
    char BRFILE[14],          //browse file
        HFILE[14],          //header file
        QFILE[13];          //query file

    char c_usernum[3];        //user number converted to character for concatenation
                                //in file name

    CLIENT_TKFAR *clnt;       /** handle to client **/

    FILE *dfile,*hfile;      /** file ptrs to server data files **/
    FILE *qf, *ef;           /** pointer to query file **/

    HCURSOR hCur;           /** cursor handle **/

    HWND hEIN, hYear;        /** handles to IDMF_BrEIN and IDMF_BrYear **/
    HWND hMRN;               // handle to IDMF_BrStart

    int cres;                /** clnt_control result **/
    int usernum, brrpcres, brres; /** user number, rpc call result, and header call result **/

    struct query brquery;    /** structure to hold header query parameters **/
    struct timeval timeout;   /** timeval structure for clnt_control **/

    //build file names using current user number
    strcpy(BRFILE, "d:brow");
    strcat(BRFILE, itoa(USER, c_usernum, 10) );
    strcat(BRFILE, ".txt\0");

    strcpy(HFILE, "d:hdr");
    strcat(HFILE, itoa(USER, c_usernum, 10) );
    strcat(HFILE, ".txt\0");

    strcpy(QFILE, "d:query");
    strcat(QFILE, itoa(USER, c_usernum, 10) );
    strcat(QFILE, ".txt\0");

    switch(message)
    {
        case WM_INITDIALOG:
            //initialize current stat structure
            init_current();

            //increment # of browse report selections
            aggregate.tot_browse_report++;

            //store start browse time
            current.start_browse = time(NULL);

            return BLD_BrowseEntryDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            switch(wParam)
```

```

{
case IDOK:
    BrYear[0] = 0;
    BrEIN[0] = 0;
    BrEstab[0] = 0;
    BrStart[0] = 0;          //starting MRN
    GetDlgItemText(hDlg, IDMF_BrYear, (LPSTR)BrYear, 5); /** get N+1 characters **
/
    GetDlgItemText(hDlg, IDMF_BrEIN, (LPSTR)BrEIN, 11);
    GetDlgItemText(hDlg, IDMF_BrEstab, (LPSTR)BrEstab, 5);
    GetDlgItemText(hDlg, IDMF_BrStart, (LPSTR)BrStart, 12);

    if (strcmp(BrYear, "1991") != 0)          /** validate the year **
/
    {
        BLD_Year_ErrDlgFunc(hDlg, message, wParam, lParam);
        hYear = GetDlgItem(hDlg, IDMF_BrYear);
        SetFocus(hYear);
    }
    else          //year OK, send header request
    {
        usernum = USER;
        /** send the header query info to a file and let the server know **/
        qf = fopen(QFILE, "wb");
        if (qf == NULL)          //if file not available drop out of all if-else
            BLD_QueryTxtDlgFunc(hDlg, message, wParam, lParam);
        else          //if file available continue
        {
            strcpy(brquery.Year, BrYear);
            strcpy(brquery.EIN, BrEIN);
            strcpy(brquery.Estab, BrEstab);
            strcpy(brquery.seq_no, DEF_SEQ_NO);
            brquery.FName[0] = 0;
            brquery.LName[0] = 0;
            brquery.SSN[0] = 0;
            strcpy(brquery.offset, BrStart);
            fwrite(&brquery, sizeof(struct query), 1, qf);
            fclose(qf);
            hCur = SetCursor (LoadCursor (NULL, IDC_WAIT));          //change to hourgl
ass
            ShowCursor (TRUE);          //show hourglass

            /** clnt_create is one step down from a straight rpc call
                it is required here for control of the time out value **/
            clnt = clnt_create(HOST, GET_SEQ_HEADER, VERSNUM, UDP);
            if(clnt == NULL)
            {
                ef = fopen(EFILE, "a");
                if (ef == NULL) //if error file not available alert user
                    BLD_ErrorFileDlgFunc(hDlg, message, wParam, lParam);
                else          //if error file available continue
                {
                    fprintf(ef, "\n%s\n", "Browse, Get Seq Header, clnt_create");
                    fprintf(ef, "%s\n", "CLIENT HANDLE IS NULL");
                    fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
                    fprintf(ef, "%d = T_ERRNO\n", t_errno);
                    fclose(ef);
                    BLD_QueryErrDlgFunc(hDlg, message, wParam, lParam);
                    ShowCursor (FALSE);          //hide hourglass
                    SetCursor (hCur);          //reset to arrow
                    hYear = GetDlgItem(hDlg, IDMF_BrYear);
                    SetFocus(hYear);
                } //end of ef != NULL
            } //end of if clnt = NULL

```

```

else          /** set re-try timeout value for employer header request**
/
{
    timeout.tv_sec = RETRY_TIME;
    timeout.tv_usec = 0;
    cres = clnt_control(clnt, CLSET_RETRY_TIMEOUT, (char _TKFAR *)&timeout
);
    if (cres == 0)
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg, message, wParam, lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Browse, Get Seq Header, clnt_control");
            fprintf(ef, "%s\n\n", "RE-TRY TIMEOUT WAS NOT SET");
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg, message, wParam, lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
            hYear = GetDlgItem(hDlg, IDMF_BrYear);
            SetFocus(hYear);
        } //end of ef != NULL
    } //end of if cres = 0
    else          /** set total timeout & request employer header info **/
    {
        timeout.tv_sec = TOT_TIME;
        timeout.tv_usec = 0;

        brrpcres = clnt_call(clnt, PROCNUM, (xdrproc_t)xdr_int,
            (caddr_t)&usernum, (xdrproc_t)xdr_int, (caddr_t)&brres, timeou
t);

        if ((brrpcres != 0) || (brres != 1)) /** check for errors **/
        {
            ef = fopen(EFILE, "a");
            if (ef == NULL) //if error file not available alert user
                BLD_ErrorFileDlgFunc(hDlg, message, wParam, lParam);
            else //if error file available continue
            {
                fprintf(ef, "\n%s\n", "Browse, Get Seq Header, clnt_call");
                fprintf(ef, "BrRPCRES = %d\n", brrpcres);
                fprintf(ef, "BrRES = %d\n", brres);
                if (brrpcres != 0) /** if rpc call failed **/
                {
                    /** rpc_createerr is a global variable returned by rpc_call
                        and relates the status of the call itself **/
                    /** t_errno further delineates the error in certain settings
**/

                    fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
                    fprintf(ef, "%d = T_ERRNO\n\n", t_errno);
                    BLD_QueryErrDlgFunc(hDlg, message, wParam, lParam);
                    hYear = GetDlgItem(hDlg, IDMF_BrYear);
                    SetFocus(hYear);
                } //end of rpc call failed
            else //if EIN could not be found by search engine
            {
                fprintf(ef, "%s\n\n", "SEARCH ENGINE COULD NOT FIND EIN");
                BLD_EINErrDlgFunc(hDlg, message, wParam, lParam);
                hEIN = GetDlgItem(hDlg, IDMF_BrEIN);
                SetFocus(hEIN);
            } //end of incorrect EIN
            fclose(ef);
        } //end of if ef != NULL
    } //end of error check

```



```

else //Get Header complete, request browse report
{
    /** clnt_create is one step down from a straight rpc call
        it is required here for control of the time out value **/
    clnt = clnt_create(HOST, BROWSE_REPORT, VERSNUM, UDP);
    if(clnt == NULL)
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Browse, Browse Report, clnt_create");
            fprintf(ef, "%s\n", "CLIENT HANDLE IS NULL");
            fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
            fprintf(ef, "%d = T_ERRNO\n", t_errno);
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
            hYear = GetDlgItem(hDlg,IDMF_BrYear);
            SetFocus(hYear);
        } //end of ef != NULL
    } //end of if clnt = NULL
    else /** set re-try timeout value for browse report request**/
    {
        timeout.tv_sec = RETRY_TIME;
        timeout.tv_usec = 0;
        cres = clnt_control(clnt,CLSET_RETRY_TIMEOUT,(char _TKFAR *)&t
imeout);

        if (cres == 0)
        {
            ef = fopen(EFILE, "a");
            if (ef == NULL) //if error file not available alert user
                BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
            else //if error file available continue
            {
                fprintf(ef, "\n%s\n", "Browse, Browse Report, clnt_control
");

                fprintf(ef, "%s\n", "RE-TRY TIMEOUT WAS NOT SET");
                fclose(ef);
                BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
                ShowCursor (FALSE); //hide hourglass
                SetCursor (hCur); //reset to arrow
                hYear = GetDlgItem(hDlg,IDMF_BrYear);
                SetFocus(hYear);
            } //end of ef != NULL
        } //end of if cres = 0
        else /** set total timeout & request browse report info
        **/

        {
            timeout.tv_sec = TOT_TIME;
            timeout.tv_usec = 0;

            brrpcres = clnt_call(clnt,PROCNUM,(xdrproc_t)xdr_int,
                (caddr_t)&usernum,(xdrproc_t)xdr_int, (caddr_t)&brres, ti
meout);

            if ((brrpcres != 0) || (brres != 1)) /** check for erro
            rs **/

            {
                ef = fopen(EFILE, "a");
                if (ef == NULL) //if error file not available alert user
                    BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
            }
        }
    }
}

```

```

else    //if error file available continue
{
    fprintf(ef, "\n%s\n", "Browse, Browse Report, clnt_call"
);
    fprintf(ef, "BrRPCRES = %d\n", brrpcres);
    fprintf(ef, "BrRES = %d\n", brres);
    if (brrpcres != 0)    /** if rpc call failed **/
    {
        /** rpc_createrr is a global variable returned by rpc_
call
and relates the status of the call itself **/
        /** t_errno further delineates the error in certain set
tings **/

        fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
        fprintf(ef, "%d = T_ERRNO\n\n", t_errno);
        BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
        hYear = GetDlgItem(hDlg,IDMF_BrYear);
        SetFocus(hYear);
    } //end of rpc call failed
    else //if MRN could not be found by search engine
    {
        fprintf(ef, "%s\n\n", "SEARCH ENGINE COULD NOT FIND MR
N");
        BLD_MRNErrDlgFunc(hDlg,message,wParam,lParam);
        hMRN = GetDlgItem(hDlg,IDMF_BrStart);
        SetFocus(hMRN);
    } //end of incorrect MRN
    fclose(ef);
} //end of if ef != NULL
} //end of error check
else    //parameters sent and data written
{
    EndDialog(hDlg,1);
    hfil = fopen(HFILE, "rb");
    dfil = fopen(BRFILE, "rb");
    if ((dfil == NULL) || (hfil == NULL))
        BLD_MissingFileDlgFunc(hDlg,message,wParam,lParam);
    else
    {
        fclose(dfil);
        fclose(hfil);
        BLD_BrowseReportDlgFunc(hDlg,message,wParam,lParam);
    }
} //everything executed properly
} //end of set total timeout and browse report request
} //end of set retry timeout for browse report request
} //end of get header complete, request browse report info
} //end of set total timeout and request header
} //end of set retry timeout for get header
} //end of if qf != NULL
} //end of year OK, send header request

break; //end of IDOK

case IDCANCEL:
    if (!BLD_BrowseEntryDlgDefault(hDlg,message,wParam,lParam))
        EndDialog(hDlg,IDCANCEL);
    break;
default:
    return BLD_BrowseEntryDlgDefault(hDlg,message,wParam,lParam);
    break;
}

```





```
int index;                /** index of list box entry **/
int tabs[] = {52,72,84,123,162,197,243,297,427}; /** tab settings for list box **/
int usernum;              /** user number **/

long int lbitems;         /** number of items in the list box **/
long num;                 /** was anything read from the file **/

struct query dquery;      /** structure to hold detail query paramet
ers **/
struct timeval timeout;    /** timevalue structure for clnt_control *
*/

struct W2Browse *pBrowse; /** pointer to a structure **/

unsigned long offset;      /** holds converted record location **/

WORD numtabs = 9;         /** number of tabs in list box **/
                          /** added 6/19/92 by LLD **/

//build file names using current user number
strcpy(BRFILE, "d:brow");
strcat(BRFILE, itoa(USER, c_usernum, 10) );
strcat(BRFILE, ".txt\0");

strcpy(DFILE, "d:det1");
strcat(DFILE, itoa(USER, c_usernum, 10) );
strcat(DFILE, ".txt\0");

strcpy(HFILE, "d:hdr");
strcat(HFILE, itoa(USER, c_usernum, 10) );
strcat(HFILE, ".txt\0");

strcpy(QFILE, "d:query");
strcat(QFILE, itoa(USER, c_usernum, 10) );
strcat(QFILE, ".txt\0");

switch(message)
{
    case WM_INITDIALOG:    /**file availability checked in BrowseEntry dialog
                          /**before this dialog is initialized

        hfil = fopen(HFILE, "rb");
        if (hfil == NULL) /**if file pointer null alert user
        {
            BLD_NULLPtrDlgFunc(hDlg,message,wParam,lParam);
            EndDialog(hDlg,1);
            return TRUE;
        }
        else /**if file availabe continue
        {
            fread(&CurrEmprInfo, sizeof(struct W2EmprInfo), 1, hfil);
            fclose(hfil);
            CreateHeaderString();
            SetDlgItemText(hDlg, ID_BrEmprHeader, HeaderString);
            SendDlgItemMessage(hDlg, IDLB_BrMatch, LB_SETHORIZONTALEXTENT, 1000, 0);
            SendDlgItemMessage(hDlg, IDLB_BrMatch, LB_SETTABSTOPS, numtabs,
                (LONG) (LPINT) tabs);

            //offcount = 0; //initialize counter for entries in list box
            brfil = fopen(BRFILE, "rb");
            pBrowse = &Browse;
```

```

if (brfil == NULL) //if file pointer null alert user
{
    BLD_NULLPtrDlgFunc(hDlg,message,wParam,lParam);
    EndDialog(hDlg,1);
    return TRUE;
}
else //if file available continue
{
    for (count=1; count <= MAXBRLB; count++) //** read from the file **/
    {
        //offcount++; //set counter
        num = fread(pBrowse, sizeof(struct W2Browse), 1, brfil);

        if(num != 0) //if file contains data read
        {
            offset = pBrowse->record_loc[0]; //** convert the **/
            offset = offset<< 8; //** record location **/
            offset = offset + pBrowse->record_loc[1]; //** because of **/
            offset = offset<<8; //** different byte **/
            offset = offset + pBrowse->record_loc[2]; //** ordering on **/
            offset = offset<<8; //** the SUN **/
            offset = offset + pBrowse->record_loc[3];

            /** display the browse info in the list box **/
            ListBoxString[0] = 0;
            sprintf(ListBoxString,"%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s",
                Browse.MRN,
                Browse.seq_no,
                Browse.wage_type,
                Browse.AnnFICAWages,
                Browse.AnnFICATips,
                Browse.FICATaxWheld,
                Browse.AnnWgsTpsOther,
                Browse.EmpSSN,
                Browse.EmpName,
                ultoa(offset, reoffset, 10));
            SendDlgItemMessage(hDlg,IDLB_BrMatch,LB_ADDSTRING,0,
                (LONG) (LPSTR)ListBoxString);
        } //end of if num != 0
    } //end of for loop - reading from the file

    fclose(brfil);
    SendDlgItemMessage(hDlg,IDLB_BrMatch,LB_SETCURSEL, (WPARAM)0, (LONG)0);
    return BLD_BrowseReportDlgDefault(hDlg,message,wParam,lParam);
} //end of if brfil != NULL
} // end of if hfil != NULL
break; //end of WM_INITDIALOG

case WM_COMMAND:
    switch(wParam)
    {
        case IDOK: //** when user hits return inside the listbox **/

            hCur = SetCursor (LoadCursor (NULL, IDC_WAIT)); //change to hourglass
            ShowCursor (TRUE); //show hourglass

            ListBoxString[0] = 0; //** use index to get string then

            index = (WORD) SendDlgItemMessage(hDlg, //** parse out the offset **/
                IDLB_BrMatch,LB_GETCURSEL,0,0); //** and the sequence number **/
            SendDlgItemMessage(hDlg,IDLB_BrMatch,LB_GETTEXT,
                index, (LONG) (LPSTR)ListBoxString);

```

\*/

```

strncpy(recoffset, ListBoxString + 95, 29);
recoffset[29] = 0;                                /** terminate string **/
strncpy(dquery.seq_no, ListBoxString + 12, 3);
dquery.seq_no[3] = 0;                              /** terminate string **/

usernum = USER;
/** send the detail query info to a file and let the server know **/
qf = fopen(QFILE, "wb");
if (qf == NULL) //if file not available drop out of all if-else
    BLD_QueryTxtDlgFunc(hDlg,message,wParam,lParam);
else //if file available continue
{
    strcpy(dquery.Year, BrYear);
    strcpy(dquery.EIN, BrEIN);
    strcpy(dquery.Estab, BrEstab);
    dquery.FName[0] = 0;
    dquery.LName[0] = 0;
    dquery.SSN[0] = 0;
    strcpy(dquery.offset, recoffset);
    fwrite(&dquery, sizeof(struct query), 1, qf);
    fclose(qf);

    /** clnt_create is one step down from a straight rpc call
        it is required here for control of the time out value **/
    clnt = clnt_create(HOST, GET_EMPL_DETAIL, VERSNUM, UDP);
    if(clnt == NULL)
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Browse, Get_Empl_Detail, clnt_create");
            fprintf(ef, "%s\n", "CLIENT HANDLE IS NULL");
            fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
            fprintf(ef, "%d = T_ERRNO\n", t_errno);
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
            hLB=GetDlgItem(hDlg, IDLB_BrMatch);
            SetFocus(hLB);
        } //end of ef != NULL
    } //end of if clnt = NULL
else //** set re-try timeout value **/
{
    timeout.tv_sec = RETRY_TIME;
    timeout.tv_usec = 0;
    cres = clnt_control(clnt,CLSET_RETRY_TIMEOUT,(char _TKFAR *)&timeout);
    if (cres == 0)
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Browse, Get_Empl_Detail, clnt_control");
            fprintf(ef, "%s\n\n", "RE-TRY TIMEOUT WAS NOT SET");
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
        }
    }
}

```



```

        SetFocus(hLB);
    } //end of ef != NULL
} //end of if cres = 0
else /** set total timeout & request employee detail */
{
    timeout.tv_sec = TOT_TIME;
    timeout.tv_usec = 0;
    drpcres = clnt_call(clnt, PROCNUM, (xdrproc_t)xdr_int,
                      (caddr_t)&usernum, (xdrproc_t)xdr_int,
                      (caddr_t)&dres, timeout);
    clnt_destroy(clnt);
    if ((drpcres != 0) || (dres != 1))
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg, message, wParam, lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Browse, Get_Empl_Detail, clnt_call");
            fprintf(ef, "%d = DRPCRES\n", drpcres);
            fprintf(ef, "%d = DRES\n", dres);
            fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
            fprintf(ef, "%d = T_ERRNO\n", t_errno);
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg, message, wParam, lParam);
            ShowCursor(FALSE); //hide hourglass
            SetCursor(hCur); //reset to arrow
            hLB = GetDlgItem(hDlg, IDLB_BrMatch);
            SetFocus(hLB);
        } //end of ef != NULL
    } //end of error check on drpcres and dres
else //parameters, communications & data verified
{
    dfil = fopen(DFILE, "rb"); /** open the detail file */
    if (dfil == NULL) //check file ptr
    {
        BLD_DFileErrDlgFunc(hDlg, message, wParam, lParam);
        ShowCursor(FALSE); //hide hourglass
        SetCursor(hCur); //reset to arrow
        hLB = GetDlgItem(hDlg, IDLB_BrMatch);
        SetFocus(hLB);
    }
else
    {
        /** read in employee detail info */
        fread(&EDetail, sizeof(struct W2EmpInfo), 1, dfil);
        fclose(dfil);
        CreateEDetail();
        BLD_EmployeeDetailDlgFunc(hDlg, message, wParam, lParam);
        ShowCursor(FALSE); //hide hourglass
        SetCursor(hCur); //reset to arrow
        hLB = GetDlgItem(hDlg, IDLB_BrMatch);
        SetFocus(hLB);
    }
} //everything executed properly
} //end of set total timeout
} //end of set re-try timeout value
} //end of qf != NULL
break; //end of IDOK

case IDCANCEL:
    if (!BLD_BrowseReportDlgDefault(hDlg, message, wParam, lParam))
        EndDialog(hDlg, IDCANCEL);
    break;

```

```
case IDLB_BrMatch:      /** when user double clicks inside the listbox **/

if (HIWORD(lParam)==LBN_DBLCLK)
{
    hCur = SetCursor (LoadCursor (NULL, IDC_WAIT)); //change to hourglass
    ShowCursor (TRUE);                               //show hourglass

    ListBoxString[0] = 0;                            /** use index to get string then
**/

    index = (WORD) SendDlgItemMessage(hDlg,           /** parse out the offset **/
        IDLB_BrMatch, LB_GETCURSEL, 0, 0);           /** and the sequence number **/
    SendDlgItemMessage(hDlg, IDLB_BrMatch, LB_GETTEXT,
        index, (LONG) (LPSTR) ListBoxString);
    strncpy(recoffset, ListBoxString + 95, 29);
    recoffset[29] = 0;                               /** terminate string **/
    strncpy(dquery.seq_no, ListBoxString + 12, 3);
    dquery.seq_no[3] = 0;                             /** terminate string **/

    usernum = USER;
    /** send the detail query info to a file and let the server know **/
    qf = fopen(QFILE, "wb");
    if (qf == NULL) //if file not available drop out of all if-else
        BLD_QueryTxtDlgFunc(hDlg, message, wParam, lParam);
    else //if file available continue
    {
        strcpy(dquery.Year, BrYear);
        strcpy(dquery.EIN, BrEIN);
        strcpy(dquery.Estab, BrEstab);
        dquery.FName[0] = 0;
        dquery.LName[0] = 0;
        dquery.SSN[0] = 0;
        strcpy(dquery.offset, recoffset);
        fwrite(&dquery, sizeof(struct query), 1, qf);
        fclose(qf);

        /** clnt_create is one step down from a straight rpc call
            it is required here for control of the time out value **/
        clnt = clnt_create(HOST, GET_EMPL_DETAIL, VERSNUM, UDP);
        if(clnt == NULL)
        {
            ef = fopen(EFILE, "a");
            if (ef == NULL) //if error file not available alert user
                BLD_ErrorFileDlgFunc(hDlg, message, wParam, lParam);
            else //if error file available continue
            {
                fprintf(ef, "\n%s\n", "Browse, Get_Empl_Detail, clnt_create");
                fprintf(ef, "%s\n", "CLIENT HANDLE IS NULL");
                fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
                fprintf(ef, "%d = T_ERRNO\n", t_errno);
                fclose(ef);
                BLD_QueryErrDlgFunc(hDlg, message, wParam, lParam);
                ShowCursor (FALSE); //hide hourglass
                SetCursor (hCur); //reset to arrow
                hLB=GetDlgItem(hDlg, IDLB_BrMatch);
                SetFocus(hLB);
            } //end of ef != NULL
        } //end of if clnt = NULL
    else //** set re-try timeout value **/
    {
        timeout.tv_sec = RETRY_TIME;
        timeout.tv_usec = 0;
        cres = clnt_control(clnt, CLSET_RETRY_TIMEOUT, (char _TKFAR *)&timeout);
    }
}
```

```
if (cres == 0)
{
    ef = fopen(EFILE, "a");
    if (ef == NULL) //if error file not available alert user
        BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
    else //if error file available continue
    {
        fprintf(ef, "\n%s\n", "Browse, Get_Empl_Detail, clnt_control");
        fprintf(ef, "%s\n\n", "RE-TRY TIMEOUT WAS NOT SET");
        fclose(ef);
        BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
        ShowCursor (FALSE); //hide hourglass
        SetCursor (hCur); //reset to arrow
        hLB=GetDlgItem(hDlg, IDLB_BrMatch);
        SetFocus(hLB);
    } //end of ef != NULL
} //end of if cres = 0
else /** set total timeout & request employe detail info**/
{
    timeout.tv_sec = TOT_TIME;
    timeout.tv_usec = 0;
    drpcres = clnt_call(clnt,PROCNUM,(xdrproc_t)xdr_int,
                      (caddr_t)&usernum, (xdrproc_t)xdr_int,
                      (caddr_t)&dres, timeout);

    clnt_destroy(clnt);
    if ((drpcres != 0) || (dres != 1))
    {
        ef = fopen(EFILE, "a");
        if (ef == NULL) //if error file not available alert user
            BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
        else //if error file available continue
        {
            fprintf(ef, "\n%s\n", "Browse, Get_Empl_Detail, clnt_call");
            fprintf(ef, "%d = DRPCRES\n", drpcres);
            fprintf(ef, "%d = DRES\n\n", dres);
            fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
            fprintf(ef, "%d = T_ERRNO\n\n", t_errno);
            fclose(ef);
            BLD_QueryErrDlgFunc(hDlg,message,wParam,lParam);
            ShowCursor (FALSE); //hide hourglass
            SetCursor (hCur); //reset to arrow
            hLB=GetDlgItem(hDlg, IDLB_BrMatch);
            SetFocus(hLB);
        } //end of ef != NULL
    } //end of error check on drpcres and dres
else //parameters, communications & data verified
{
    dfil = fopen(DFILE, "rb"); /** open the detail file **/
    if (dfil == NULL) //check file ptr
    {
        BLD_DFileErrDlgFunc(hDlg,message,wParam,lParam);
        ShowCursor (FALSE); //hide hourglass
        SetCursor (hCur); //reset to arrow
        hLB=GetDlgItem(hDlg, IDLB_BrMatch);
        SetFocus(hLB);
    }
else
{
    /** read in employe info **/
    fread(&EDetail, sizeof(struct W2EmpInfo), 1, dfil);
    fclose(dfil);
    CreateEDetail();
    BLD_EmployeeDetailDlgFunc(hDlg,message,wParam,lParam);
    ShowCursor (FALSE); //hide hourglass
    SetCursor (hCur); //reset to arrow
}
```



```
        hLB=GetDlgItem(hDlg, IDLB_BrMatch);
        SetFocus(hLB);
    }
    } //everything executed properly
    } //end of set total timeout
    } //end of set re-try timeout value
    } //end of qf != NULL
    } //end of if double-clicked
break; //end of IDLB_BrMatch

case ID_CLOSE:
    current.stop_browse = time(NULL); //store browse report stop time
    fill_current(); //calculate net browse report time
    fill_aggregate(); //adjust browse totals and averages
    write_aggregate(hDlg,message,wParam,lParam); //write aggregate stats to file
    EndDialog(hDlg,ID_CLOSE);
    break;

case ID_BrAddRec: //when user selects additional records
                  //file availability previously checked

    aggregate.tot_add_records++; //increment stats for add records
    hCur = SetCursor (LoadCursor (NULL, IDC_WAIT)); //change to hourglass
    ShowCursor (TRUE); //show hourglass

    lbitems = SendDlgItemMessage(hDlg,IDLB_BrMatch,LB_GETCOUNT,0,0);

    brfil = fopen(BRFILE, "rb");
    if (brfil == NULL)
    {
        BLD_DataErrDlgFunc(hDlg,message,wParam,lParam);
        ShowCursor (FALSE); //hide hourglass
        SetCursor (hCur); //reset to arrow
        hLB=GetDlgItem(hDlg, IDLB_BrMatch);
        SetFocus(hLB);
    }
    else
    {
        pBrowse = &Browse;
        fseek(brfil, sizeof(struct W2Browse)*lbitems, 0); //position file pointer

        for (count=1; count <=MAXBRLB; count++) //** read from the file **/
        {
            num = fread(pBrowse, sizeof(struct W2Browse), 1, brfil);

            if(num != 0) //if not EOF
            {
                offset = pBrowse->record_loc[0]; //** convert the **/
                offset = offset<< 8; //** offset **/
                offset = offset + pBrowse->record_loc[1]; //** because of **/
                offset = offset<<8; //** different byte **/
                offset = offset + pBrowse->record_loc[2]; //** ordering on **/
                offset = offset<<8; //** the SUN **/
                offset = offset + pBrowse->record_loc[3];

                /** display the browse info in the list box **/
                ListBoxString[0] = 0;
                sprintf(ListBoxString,"%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s",
                    Browse.MRN,
                    Browse.seq_no,
                    Browse.wage_type,
                    Browse.AnnFICAWages,
```

```

        Browse.AnnFICATips,
        Browse.FICATaxWheld,
        Browse.AnnWgsTpsOther,
        Browse.EmpSSN,
        Browse.EmpName,
        ultoa(offset, reoffset, 10));
    if ((SendDlgItemMessage(hDlg, IDLB_BrMatch, LB_ADDSTRING, 0,
        (LONG) (LPSTR) ListBoxString)) == LB_ERRSPACE)
    {
        aggregate.tot_add_records--; //adjust count when listbox full
        count = MAXBRLB + 1;        //and user presses add_records in er
ror
    }
    } //end of if not EOF
} //end of for loop
fclose(brfil);
} //end of if brfil != NULL
ShowCursor (FALSE); //hide hourglass
SetCursor (hCur); //reset to arrow
hLB = GetDlgItem(hDlg, IDLB_BrMatch);
SetFocus(hLB);
SendDlgItemMessage(hDlg, IDLB_BrMatch, LB_SETCURSEL,
    (WORD) lbitems, (LONG) 0);
break; //end of ID_BrAddRec

default:
    return BLD_BrowseReportDlgDefault (hDlg, message, wParam, lParam);
    break;
}
break;

default:
    return BLD_BrowseReportDlgDefault (hDlg, message, wParam, lParam);
    break;
}
return TRUE; /* Did process the message */
} //END OF DISPLAY BROWSE REPORT RECORDS DIALOG

/*****
/* prompts user to enter "printing report" info */
*****/

int BLD_PrintDlgFunc (hWnd, message, wParam, lParam) /* Startup procedure for modal dialog box
*/
HWND hWnd;
UINT message;
WPARAM wParam;
LONG lParam;
{
    FARPROC lpProc;
    int ReturnValue;

    lpProc = MakeProcInstance (BLD_PrintDlgProc, hInst);
    ReturnValue = DialogBox (hInst, (LPSTR) "PRINT", hWnd, lpProc);
    FreeProcInstance (lpProc);
    if (ReturnValue == -1)
        BLDDisplayMessage (hWnd, BLD_CannotCreate, "PRINT",
            MB_OK | MB_ICONHAND);
    return ReturnValue;
}

```

```
BOOL FAR PASCAL BLD_PrintDlgProc(hDlg, message, wParam, lParam) /* Modal dialog box procedure */
{
    HWND hDlg;
    UINT message;
    WPARAM wParam;
    LONG lParam;

    char QFILE[13];          //query file

    char c_usernum[3];       //user number converted to character
                             //for concatenation in file name

    FILE *qf, *ef;          /** pointer to query file and error file**/

    HCURSOR hCur;           /** cursor handle **/

    HWND hEIN, hYear;        /** handle to IDMF_PEIN and IDMF_PYear **/
    HWND hPW;                /** handle to IDMF_Password child window **/

    int usernum, prpcres, pres; /** user number, rpc call result, and header call result */
}

struct query pquery;        /** structure to hold header query parameters **/

//build file name using current user number
strcpy(QFILE, "d:query");
strcat(QFILE, itoa(USER, c_usernum, 10) );
strcat(QFILE, ".txt\0");

switch(message)
{
    case WM_INITDIALOG:
        //increment # of times print report selected
        aggregate.tot_print_report++;

        return BLD_PrintDlgDefault(hDlg,message,wParam,lParam);
        break;

    case WM_COMMAND:
        switch(wParam)
        {
            case IDOK:
                PW[0] = 0;
                PYear[0] = 0;
                PEIN[0] = 0;
                PEstab[0] = 0;
                PSeq[0] = 0;
                GetDlgItemText(hDlg, IDMF_Password, (LPSTR)PW, 6);
                GetDlgItemText(hDlg, IDMF_PYear, (LPSTR)PYear, 5);  /** get N+1 characters **

                GetDlgItemText(hDlg, IDMF_PEIN, (LPSTR)PEIN, 11);
                GetDlgItemText(hDlg, IDMF_PEstab, (LPSTR)PEstab, 5);
                GetDlgItemText(hDlg, IDMF_PSeq, (LPSTR)PSeq, 4);

                if ((strcmp(PASSWORD, PW)) != 0)          //check password
                {
                    BLD_PWErrDlgFunc(hDlg,message,wParam,lParam);
                }
            }
        }
}
```



```

    hPW = GetDlgItem(hDlg, IDMF_Password);
    SetFocus(hPW);
}
else //password OK, check year
if (strcmp(PYear, "1991") != 0)
{
    BLD_Year_ErrDlgFunc(hDlg,message,wParam,lParam);
    hYear = GetDlgItem(hDlg,IDMF_PYear);
    SetFocus(hYear);
}
else //year OK, get header info
{
    usernum = USER;
    /** send the header query info to a file and let the server know **/
    qf = fopen(QFILE, "wb");
    if (qf == NULL) //if file not available drop out of all if-else
        BLD_QueryTxtDlgFunc(hDlg,message,wParam,lParam);
    else //if file available continue
    {
        hCur = SetCursor (LoadCursor (NULL, IDC_WAIT)); //change to hourglass

        ShowCursor (TRUE); //show hourglass
        strcpy(pquery.Year, PYear);
        strcpy(pquery.EIN, PEIN);
        strcpy(pquery.Estab, PEstab);
        strcpy(pquery.seq_no, PSeq);
        pquery.FName[0] = 0;
        pquery.LName[0] = 0;
        pquery.SSN[0] = 0;
        pquery.offset[0] = 0;
        fwrite(&pquery, sizeof(struct query), 1, qf);
        fclose(qf);
        prpcres = rpc_call(HOST,PRINT_REPORT,VERSNUM,
            PROCNUM,(xdrproc_t)xdr_int,
            (char _TKFAR *)&usernum,(xdrproc_t)xdr_int, (char _TKFAR *)&pres,"
visible");

        if ((prpcres != 0) || (pres != 1)) //check for errors
        {
            ef = fopen(EFILE, "a");
            if (ef == NULL) //if error file not available alert user
                BLD_ErrorFileDlgFunc(hDlg,message,wParam,lParam);
            else //if error file available continue
            {
                fprintf(ef, "\n%s\n", "Print, Print Report, rpc_call");
                fprintf(ef, "%d = PRPCRES\n", prpcres);
                fprintf(ef, "%d = PRES\n", pres);
                if (prpcres != 0) //if rpc call fails
                {
                    /** rpc_createerr is a global variable returned by
                        rpc_call and relates the status of the call itself **/
                    /** t_errno further delineates the error in certain
                        settings **/
                    fprintf(ef, "%d = RPC ERROR\n", rpc_createerr);
                    fprintf(ef, "%d = T_ERRNO\n\n", t_errno);
                    BLD_SysErrDlgFunc(hDlg,message,wParam,lParam);
                    ShowCursor (FALSE); //hide hourglass
                    SetCursor (hCur); //reset to arrow
                    hPW = GetDlgItem(hDlg, IDMF_Password);
                    SetFocus(hPW);
                }
            }
            else //if EIN or seq no. could not be found
            {
                fprintf(ef, "%s\n", "SEARCH ENGINE COULD NOT FIND EIN or SEQ. N
0.");

```

```

        fprintf(ef, "%s\n\n", "OR REPORT CONTAINS MORE THAN 5000 EMPLOY
EES");

```

```

        BLD_EINorSeqErrDlgFunc(hDlg,message,wParam,lParam);
        ShowCursor (FALSE);           //hide hourglass
        SetCursor (hCur);             //reset to arrow
        hEIN = GetDlgItem(hDlg,IDMF_PEIN);
        SetFocus(hEIN);
    }
    fclose(ef);
    } //end of if rpc call fails
    } //end of ef != NULL
else //everything OK, print report
{
    ShowCursor (FALSE);           //hide hourglass
    SetCursor (hCur);             //reset to arrow
    EndDialog(hDlg,1);
    aggregate.tot_pr_report++; //increment # of times report printed

    //write aggregate stats to file
    write_aggregate(hDlg,message,wParam,lParam);
    BLD_Function5DlgFunc(hDlg,message,wParam,lParam);
}
} //end of if qf != NULL
} //end of year OK, get header info
break; // end of IDOK

```

```

case IDCANCEL:
    if (!BLD_PrintDlgDefault(hDlg,message,wParam,lParam))
        EndDialog(hDlg,IDCANCEL);
    break;

```

```

default:
    return BLD_PrintDlgDefault(hDlg,message,wParam,lParam);
    break;
}
break;

```

```

default:
    return BLD_PrintDlgDefault(hDlg,message,wParam,lParam);
    break;
}

```

```

return TRUE; /* Did process the message */
} //END OF PRINT REPORT ENTRY SCREEN

```

```

/*****
/* displays EXIT dialog box */
*****/

```

```

int BLD_Function2DlgFunc(hWnd,message,wParam,lParam) /* Startup procedure for modal dialog
box */

```

```

HWND hWnd;
UINT message;
WPARAM wParam;
LONG lParam;
{
    FARPROC lpProc;
    int ReturnValue;

```

```

    lpProc = MakeProcInstance(BLD_Function2DlgProc,hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)"EXIT", hWnd, lpProc);
    FreeProcInstance(lpProc);

```

```
if (ReturnValue== -1)
    BLDDisplayMessage(hWnd,BLD_CannotCreate,"EXIT",
        MB_OK | MB_ICONHAND);
return ReturnValue;
}
```

```
BOOL FAR PASCAL BLD_Function2DlgProc(hDlg, message, wParam, lParam) /* Modal dialog box procedure */
HWND hDlg;
UINT message;
WPARAM wParam;
LONG lParam;
{
```

```
    switch(message)
    {
        case WM_INITDIALOG:
            return BLD_Function2DlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            switch(wParam)
            {
                case IDOK:
                    if (!BLD_Function2DlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;
                case IDCANCEL:
                    if (!BLD_Function2DlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;
                default:
                    return BLD_Function2DlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
            return BLD_Function2DlgDefault(hDlg,message,wParam,lParam);
            break;
    }
    return TRUE; /* Did process the message */
} //END OF EXIT DIALOG
```

```
//  DETAIL DISPLAY FUNCTIONS
```

```
/* *****
/* displays employee detail
/* *****
```

```
int BLD_EmployeeDetailDlgFunc(hWnd,message,wParam,lParam) /* Startup procedure for modal dialog box */
HWND hWnd;
UINT message;
WPARAM wParam;
LONG lParam;
{
    FARPROC lpProc;
    int ReturnValue;

    lpProc = MakeProcInstance((FARPROC)BLD_EmployeeDetailDlgProc,hInst);
```



```
ReturnValue = DialogBox(hInst, (LPSTR)"EDETAIL", hWnd, lpProc);
FreeProcInstance(lpProc);
if (ReturnValue== -1)
    BLDDisplayMessage(hWnd, BLD_CannotCreate, "EDETAIL",
                      MB_OK | MB_ICONHAND);
return ReturnValue;
}
```

```
BOOL FAR PASCAL BLD_EmployeeDetailDlgProc(hDlg, message, wParam, lParam) /* Modal dialog box procedure */
```

```
HWND hDlg;
```

```
UINT message;
```

```
WPARAM wParam;
```

```
LONG lParam;
```

```
{
    int num; /* number of copies to be printed */
```

```
switch(message)
```

```
{
    case WM_INITDIALOG:
        //increment # of times employee detail selected
        aggregate.tot_ee_detail++;

        SetDlgItemText(hDlg, ID_EDetail, EDetailString);
        return BLD_EmployeeDetailDlgDefault(hDlg, message, wParam, lParam);
        break;

    case WM_COMMAND:
        switch(wParam)
        {
            case IDOK:
                if (!BLD_EmployeeDetailDlgDefault(hDlg, message, wParam, lParam))
                    EndDialog(hDlg, IDOK);
                break;

            case IDCANCEL:
                if (!BLD_EmployeeDetailDlgDefault(hDlg, message, wParam, lParam))
                    EndDialog(hDlg, IDCANCEL);
                break;

            case IDPRINTED:
                //increment # of times print employee detail selected
                aggregate.tot_pr_ee_detail++;

                BLD_GetNumCopyDlgFunc(hDlg, message, wParam, lParam);
                num = atoi(cnum);

                //increment # of details printed
                aggregate.tot_ee_det_printed = aggregate.tot_ee_det_printed + num;

                CreatePrintEDetail();
                EndDialog(hDlg, IDPRINTED); //end current dialog
                BLD_PrintEmpDetailDlgFunc(hDlg, message, wParam, lParam);
                break;

            default:
                return BLD_EmployeeDetailDlgDefault(hDlg, message, wParam, lParam);
                break;
        }
        break;

    default:
        return BLD_EmployeeDetailDlgDefault(hDlg, message, wParam, lParam);
        break;
}

return TRUE; /* Did process the message */
```

```
} //END OF DISPLAY EMPLOYEE DETAIL DIALOG
```

```
/* ***** */
/* displays header detail info */
/* ***** */

int BLD_HeaderDetailDlgFunc(hWnd,message,wParam,lParam) /* Startup procedure for modal dia
log box */
HWND hWnd;
UINT message;
WPARAM wParam;
LONG lParam;
{
    FARPROC lpProc;
    int ReturnValue;

    lpProc = MakeProcInstance((FARPROC)BLD_HeaderDetailDlgProc,hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)"HDETAIL", hWnd, lpProc);
    FreeProcInstance(lpProc);
    if (ReturnValue== -1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,"HDETAIL",
            MB_OK | MB_ICONHAND);
    return ReturnValue;
}

BOOL FAR PASCAL BLD_HeaderDetailDlgProc(hDlg, message, wParam, lParam) /* Modal dialog box
procedure */
HWND hDlg;
UINT message;
WPARAM wParam;
LONG lParam;
{

    switch(message)
    {
        case WM_INITDIALOG:
            //increment # of times employer detail selected
            aggregate.tot_er_detail++;

            CreateHDetailString();
            SetDlgItemText(hDlg,ID_HDetail,HDetailString);
            return BLD_HeaderDetailDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            switch(wParam)
            {
                case IDOK:
                    if (!BLD_HeaderDetailDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDOK);
                    break;

                case IDCANCEL:
                    if (!BLD_HeaderDetailDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;

                case IDPRINTHD:
                    aggregate.tot_pr_er_detail++; //increment # of times empr. det. printed
                    EndDialog(hDlg,IDPRINTHD);
            }
    }
}
```

```

        BLD_PrintHeaderDetailDlgFunc(hDlg,message,wParam,lParam);
        break;

    default:
        return BLD_HeaderDetailDlgDefault(hDlg,message,wParam,lParam);
        break;
    }
    break;

default:
    return BLD_HeaderDetailDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE; /* Did process the message */
} //END OF DISPLAY HEADER DETAIL DIALOG

/*****
/* displays report totals */
*****/

int BLD_ReportTotalsDlgFunc(hWnd,message,wParam,lParam) /* Startup procedure for modal dia
log box */
HWND hWnd;
UINT message;
WPARAM wParam;
LONG lParam;
{
    FARPROC lpProc;
    int ReturnValue;

    lpProc = MakeProcInstance((FARPROC)BLD_ReportTotalsDlgProc,hInst);
    ReturnValue = DialogBox(hInst, (LPSTR)"TOTALS", hWnd, lpProc);
    FreeProcInstance(lpProc);
    if (ReturnValue==-1)
        BLDDisplayMessage(hWnd,BLD_CannotCreate,"TOTALS",
                           MB_OK | MB_ICONHAND);

    return ReturnValue;
}

BOOL FAR PASCAL BLD_ReportTotalsDlgProc(hDlg, message, wParam, lParam) /* Modal dialog box
procedure */
HWND hDlg;
UINT message;
WPARAM wParam;
LONG lParam;
{
    char EINstr[20], RPTstr[14]; //strings for displaying EIN and Report Number

    switch(message)
    {
        case WM_INITDIALOG:
            //increment # of times report totals selected
            aggregate.tot_final++;

            CreateTotalString();
            SetDlgItemText(hDlg,ID_Tot,TotalString);
            strcpy(EINstr, "EIN: ");
            strcat(EINstr, CurrEmprInfo.EIN);
            SetDlgItemText(hDlg,ID_TotEIN, EINstr);
            strcpy(RPTstr, "RPT-NO: ");

```



```

        strcat(RPTstr, CurrEmprInfo.seq_no);
        SetDlgItemText(hDlg, ID_TotRpt, RPTstr);
        return BLD_ReportTotalsDlgDefault(hDlg, message, wParam, lParam);
        break;

    case WM_COMMAND:
        switch(wParam)
        {
            case IDPRINTTOT:
                aggregate.tot_pr_final++; //increment # of times print final totals sele
cted

                EndDialog(hDlg, IDPRINTTOT);
                BLD_TotNowPrintDlgFunc(hDlg, message, wParam, lParam);
                break;

            case IDOK:
                if (!BLD_ReportTotalsDlgDefault(hDlg, message, wParam, lParam))
                    EndDialog(hDlg, IDOK);
                break;

            case IDCANCEL:
                if (!BLD_ReportTotalsDlgDefault(hDlg, message, wParam, lParam))
                    EndDialog(hDlg, IDCANCEL);
                break;

            default:
                return BLD_ReportTotalsDlgDefault(hDlg, message, wParam, lParam);
                break;
        }
        break;

    default:
        return BLD_ReportTotalsDlgDefault(hDlg, message, wParam, lParam);
        break;
}

return TRUE; /* Did process the message */
} //END OF DISPLAY REPORT TOTALS DIALOG

```

# // MISCELLANEOUS FUNCTIONS AND DIALOGS

```

/*****
/* initialization for magic fields */
*****/

```

```

BOOL BLD_ApplicationAppInit(hInst, hPrev, pCmdShow, lpCmd) /* Initialization for application
*/
HANDLE hInst; /* Handle to this application instance. */
HANDLE hPrev; /* Handle to previous instance of application. */
int *pCmdShow; /* Pointer to variable that specifies how main window is to be shown. */
LPSTR lpCmd; /* Long pointer to the command line. */
{
    if(!hPrev) MfInitMFEDIT();
    return TRUE;
} //END OF MAGIC FIELDS INITIALIZATION

```

```

/*****

```

```
/* quits the application */
/*****/

BOOL BLD_QuitFuncUDCFunc (hWnd,message,wParam,lParam) /* User Defined Code */
HWND hWnd;
UINT message;
WPARAM wParam;
LONG lParam;
{
    write_aggregate(hWnd,message,wParam,lParam); //write aggregate stats to file
    PostQuitMessage(0);
    return TRUE;
} //END OF QUIT APPLICATION FUNCTION


// appears when user presses arrow icon to change employer header
// ****
// Modal Dialog Box: SEQ
// ****

// Startup procedure for modal dialog box
int BLD_SequenceDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_SequenceDlgFuncDef(hWnd,(char *)NULL);
}


// Modal dialog box procedure
BOOL CALLBACK BLD_SequenceDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD wId;

    switch(message)
    {
        case WM_INITDIALOG:
            SetDlgItemText(hDlg,ID_SeqNo, sequence);
            return BLD_SequenceDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    GetDlgItemText(hDlg,ID_SeqNo,(LPSTR)sequence,4); /** get N+1 char **/
                    EndDialog(hDlg,IDOK);
                    break;

                case IDCANCEL:
                    if (!BLD_SequenceDlgDefault(hDlg,message,wParam,lParam))
                        EndDialog(hDlg,IDCANCEL);
                    break;

                default:
                    return BLD_SequenceDlgDefault(hDlg,message,wParam,lParam);
                    break;
            }
            break;

        default:
    }
```

```

        return BLD_SequenceDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE;// Did process the message
} //end of get new sequence number to change employer header

```

```
// appears when user presses the QP icon in single query
// displays the most recently entered query parameters
// *****
//          Modal Dialog Box: QPARAM
// *****
```

```
// Startup procedure for modal dialog box
int BLD_qparamDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_qparamDlgFuncDef(hWnd,(char *)NULL);
}
```

```
// Modal dialog box procedure
BOOL CALLBACK BLD_qparamDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    WORD        wId;

    switch(message)
    {
        case WM_INITDIALOG:
            //increment # of times qparam is selected
            aggregate.tot_qp++;

            SetDlgItemText(hDlg, qp_year, QYear);
            SetDlgItemText(hDlg, qp_ein, QEIN);
            if ( strlen(QEstab) != 0 )
                SetDlgItemText(hDlg, qp_estab, QEstab);
            else
                SetDlgItemText(hDlg, qp_estab, "NONE\0");
            if ( strlen(LName) != 0 )
                SetDlgItemText(hDlg, qp_lname, LName);
            else
                SetDlgItemText(hDlg, qp_lname, "NONE\0");
            if ( strlen(FName) != 0 )
                SetDlgItemText(hDlg, qp_fname, FName);
            else
                SetDlgItemText(hDlg, qp_fname, "NONE\0");
            if ( strlen(QSSN) != 0 )
                SetDlgItemText(hDlg, qp_ssn, QSSN);
            else
                SetDlgItemText(hDlg, qp_ssn, "NONE\0");
            return BLD_qparamDlgDefault(hDlg,message,wParam,lParam);
            break;

        case WM_COMMAND:
            wId=LOWORD(wParam);
            switch(wId)
            {
                case IDOK:
                    EndDialog(hDlg,IDOK);
                    break;

                case IDCANCEL:

```



```
        if (!BLD_qparamDlgDefault(hDlg,message,wParam,lParam))
            EndDialog(hDlg,IDCANCEL);
        break;

    default:
        return BLD_qparamDlgDefault(hDlg,message,wParam,lParam);
        break;
    }
    break;

default:
    return BLD_qparamDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE; // Did process the message
}

// *****
//          Modal Dialog Box: QUESTION
// *****

// Startup procedure for modal dialog box
int BLD_questDlgFunc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)
{
    return BLD_questDlgFuncDef(hWnd,(char *)NULL);
}

// Modal dialog box procedure
BOOL CALLBACK BLD_questDlgProc(HWND hDlg,UINT message,WPARAM wParam,LPARAM lParam)
{
    HWND          Q1Y, Q2N;          //handles to radio buttons

    UINT          check;              //whether radio button is checked, 0 = no, 1 = yes

    WORD          wId;

    switch(message)
    {
    case WM_INITDIALOG:

        Q1Y = GetDlgItem(hDlg, ID_Q1Yes);          //default for question 1 is Yes
        Q2N = GetDlgItem(hDlg, ID_Q2No);          //default for question 2 is No
        SendMessage(Q1Y, BM_SETCHECK, 1, 0L);
        SendMessage(Q2N, BM_SETCHECK, 1, 0L);
        return BLD_questDlgDefault(hDlg,message,wParam,lParam);
        break;

    case WM_COMMAND:
        wId=LOWORD(wParam);
        switch(wId)
        {
        case IDOK:

            //when OK button is pressed write question info to
            //current internal statistic structure

            check = IsDlgButtonChecked(hDlg, ID_Q1Yes);
            if (check)
                current.resolved = 1;          //record yes to question 1 (resolved)
            else
                current.resolved = 0;          //record no to question 1 (unresolved)
        }
    }
}
```

```
check = IsDlgButtonChecked(hDlg, ID_Q2No);
if (check)
    current.interrupted = 0; //record no to question 2 (not interrupted)
else
    current.interrupted = 1; //record yes to question 2 (interrupted)

    fill_current();           //calculate net times
    fill_aggregate();         //calculate averages, percentages, and totals
    write_aggregate(hDlg,message,wParam,lParam); //record statistics in file
    EndDialog(hDlg, IDOK);
    break;
case IDCANCEL:
    if (!BLD_questDlgDefault(hDlg,message,wParam,lParam))
        EndDialog(hDlg, IDCANCEL);
    break;
default:
    return BLD_questDlgDefault(hDlg,message,wParam,lParam);
    break;
}
break;

default:
    return BLD_questDlgDefault(hDlg,message,wParam,lParam);
    break;
}
return TRUE; // Did process the message
}
```

```
//whatuser.c
//created 12/15/93 by Laura L. Downey, Comp. Scientist, NIST
//reads usernum.fil and displays the user number on the console
//created using MS C 6.0 pwb

//NOTE: whatuser.exe must be located in the same directory as usernum.fil

#include <stdio.h>                //standard i/o

main()
{
    char UFILE[] = "usernum.fil"; //file containing user number

    int num;                      //was anything read from the file
    int user;                    //user number read from file

    FILE *uf;                    //file ptr to usernum.fil

    uf = fopen(UFILE, "rt");
    if (uf == NULL)
    {
        printf("UNABLE TO OPEN %s\n, PROGRAM EXITING", UFILE);
        exit(0);
    }
    else //uf != NULL
    {
        num = fscanf(uf, "%d", &user);
        if (num == 0)
        {
            printf("USERNUM.FIL WAS EMPTY, PROGRAM EXITING");
            fclose(uf);
            exit(0);
        }
        else
        {
            system("cls");
            printf("\n\nUSER NUMBER = %d\n\n", user);
            fclose(uf);
        }
    } //end of uf != NULL
} //END OF MAIN
```



## **D.2 Search Engine Code (including utilities)**



# Makefile

Thu Oct 28 13:57:12 1993

1

```

#
# Top Level Make file (~/.ssapilot/Makefile)
#
# by: Natalie Willman
#
# This Makefile is at the highest level
# level in the project heirarchy. It will
# progress to each of the subdirectories,
# perform a make (it/install/clean/bare/depend)
# using the makefile in that directory
#
# subdirectories
SUBS = src

# command list
it install clean depend bare:
    @X='pwd'; \
    echo '<<<' $$X '>>>'; \
    for i in $(SUBS); \
    do echo '<<<' \
        cd $$X/$$i; \
        make $$i; \
    done
    , $$1 '>>>'; \

```



Makefile Thu Oct 28 13:58:01 1993

1

```
#
# SRC level Make file (-/ssapllot/src/Makefile)
#
# by Natalie Willman
#
# This Makefile is at the 2nd level of the
# project heirarchy, the src directory. It
# will progress to each of the subdirectories,
# perform a make (it/install/clean/bare/depend)
# using a makefile in that directory.

# subdirectory macro
SUBS = lib bin

# command list
it install clean depend bare :
    @X='pwd'; \
    echo '<<<' $$X '>>>'; \
    for i in $(SUBS); \
    do echo '<<<' \
    cd $$X/$$i; \
    make $$i; \
    done
    , $$i '>>>'; \
```

```
# SRC/BIN Level Make file (~/.ssapilot/src/bin/Makefile)
#
# by:  Natalie Willman
#
# This Makefile is at the third level of the
# project heirarchy.  It will progress to each
# of the subdirectories, perform a make
# (it/install/clean/bare/depend) using the
# makefile in that directory

# subdirectories
SUBS = client download index indexemplr parse search addmatch \
      search_blanket search_browse search_detail search_header \
      search_print search_single debug sysadm_print fix_parse

# command list
it install clean depend bare :
    @X='pwd'; \
    echo '<<<' $$X '>>>'; \
    for i in $(SUBS); \
    do echo '<<<' \
        , $$i '>>>'; \
    cd $$X/$$i; \
    make $$i; \
    done
```

```

# SRC/LIB Level Make file (~/.ssapilot/src/lib/Makefile)
#
# by Natalie Willman
#
# This Makefile is at the third level of the
# project hierarchy. It will progress to each
# of the subdirectories, perform a make
# (lt/install/clean/bare/depend) using
# the makefile in that directory
#
# subdirectories
SUBS = general test btrees_data btrees_empl

# command list
it install clean depend bare :
    @X='pwd'; \
    echo '<<<' $$X '>>>'; \
    for i in $(SUBS); \
    do echo '<<<'; \
    cd $$X/$$i; \
    make $$i; \
    done
    , $$i '>>>'; \

```



```

/*
 * btreestruct.h
 * version 3
 * 09/08/93
 *
 * by Natalie Willman
 *
 * This is the header file for btree.c. It defines
 * the value of m which determines the order of the b+-tree,
 * the structure for the node of the btree, and the
 * structure for the linked list in the duplicate file (used
 * before the file is ordered). In addition, function prototypes
 * are given for the module btree.c.
 *
 * It is necessary that the params header file for
 * the appropriate record be included before btreestruct.h in
 * the btree.c file so that KEYLEN is defined.
 */

/* Define Statements */
#define MINCHILD 310 /* MIN # of child nodes (m=2*MINCHILD) */
#define MAXCHILD 2*MINCHILD-1 /* MAX # of keys in node (m=MAXCHILD+1) */
#ifndef TRUE
#define TRUE 1
#endif
#ifndef FALSE
#define FALSE 0
#endif

/* Function Prototypes */
struct MEMNODE *allocate_node();
struct MEMNODE *btree_create();
long search_tree();
long find_record();
struct KEYLIST *add_new();

/* Structure definition of the B+-tree Node (as stored in memory) */
struct MEMNODE
{
    short count; /* Number of keys the node has */
    char key[MAXCHILD][KEYLEN]; /* Key values for child selection */
    char leaf; /* Boolean value indicating if leaf */
    long self_offset; /* offset to this node in data file */
    long num_dupe[MAXCHILD]; /* count of duplicates for the key */
    long dupe_offset[MAXCHILD]; /* dupe offset in temp dupe file */
    union
    {
        struct KEYLIST *dup[MAXCHILD+1]; /* linked list of dupe offsets (mem) */
        struct MEMNODE *mem[MAXCHILD+1]; /* Ptrs to next child in non-leaf */
        long disk[MAXCHILD+1]; /* or record in leaf node (disk or */
        }branch; /* memory pointer) */
};

/* Structure definition of the B+-tree Node (as written to the file) */
struct NODE
{
    short count; /* Number of keys the node has */
    char key[MAXCHILD][KEYLEN]; /* Key values for child selection */
    long freq[MAXCHILD]; /* frequency of the grams in dbase */
    char leaf; /* Boolean value indicating if leaf */
};

long branch[MAXCHILD+1]; /* disk offset to children */
};

/* structure definition of the linked list used to hold dupe offsets */
/* in memory */
struct KEYLIST
{
    long offset; /* Offset to record in the browse file */
    char weight; /* weight of record (number of grams) */
    struct KEYLIST *next_key; /* Offset to next record w/ key in */
    }; /* list file */

/* structure definition of the duplicate record information on disk */
/* in the final file */
struct RECINFO
{
    long dupe_offset; /* array of offsets to dupe records */
    char dupe_weight; /* weight of record (number of grams) */
    };

/* structure definition of the structure used to hold dupe offsets */
/* in the temporary dupe file */
struct DUPELIST
{
    struct RECINFO record[MAXDUPE]; /* pointer to the next set of offsets */
    long next_set;
    };

```



```

/*
 * eamateststruct.h
 * version 3
 *
 * 09/08/93
 *
 * by Natalie Willman
 *
 * This header file defines the structure for the record of data
 * to be read from the file, the record identifier bytes for each
 * record, function prototypes for the module eamate.c, and other
 * configurable parameters.
 */

/* Function Prototypes */
int read_eamate_w2header();
int write_eamate_w2header();
void display_eamate_w2header();
void cp_eamate_w2header();
void cp_eamate_w2final();
void cp_eamate_w2cum();
int write_eamate_w2header_info();
int read_eamate_w2header_info();
void display_eamate_w2header_info();

int read_eamate_w2employee_detail();
long write_eamate_w2employee_detail();
void display_eamate_w2employee_detail();
long write_eamate_w2employee_browse();
int read_eamate_w2employee_browse();
void display_eamate_w2employee_browse();

/*
 * EAMATE W2 Employer Header Information Record (Complete)
 *
 * Consists of W2 Employer Header
 *           W2 Employer Final Total
 *           W2 Employer Cumulative Ein (if it exists)
 */
struct EAMATE_W2EMPLR_INFO
{
    /* W2 Employer Header */
    char ein[11];
    char est[5];
    char rpt_yr[5];
    char proc_yr[5];
    char tape_lib_num[7];
    char type_emplr[2];
    char name_code[2];
    char other_ein[10];
    char mrn[12];
    char end_mrn[12];
    char seq_no[SEQ_SIZE+1];

    char name[48];
    char street_add[41];
    char city[26];
    char state[11];
    char zip_code[6];

    long platter_side;
    long initials;
    long num_recs;
    long browse_start;

    /* W2 Employer Final Total */
    char proc_wages[15];
    char rep_wages[15];
    char proc_tips[14];
    char rep_tips[14];
    char proc_other[15];
    char rep_other[15];
    char proc_fed_tax[14];
    char rep_fed_tax[14];
    char rep_fed_tax[14];
    char proc_fica_tax[14];
    char rep_fica_tax[14];
    char proc_earn_inc[14];
    char rep_earn_inc[14];
    char proc_items[8];
    char rep_items[8];

    char proc_defcomp[15];
    char rep_defcomp[15];
    char proc_nonqual[15];
    char rep_nonqual[15];
    char proc_med_wages[15];
    char rep_med_wages[15];
    char proc_med_tax[15];
    char rep_med_tax[15];

    /* W2 Employer Cumulative EIN Totals */
    char cflag;
    char cproc_wages[15];
    char cproc_tips[14];
    char cproc_other[15];
    char cproc_fed_tax[14];
    char cproc_fica_tax[14];
    char cproc_earn_inc[14];
    char cproc_items[8];
};

/* EAMATE W2 Employer Header Record
 *
 * This is the structure of the Employer Header Information record
 */
struct EAMATE_W2EMPLR_HEADER
{
    char ein[11];
    char est[5];
    char rpt_yr[5];
    char proc_yr[5];
    char tape_lib_num[7];
    char type_emplr[2];
    char name_code[2];
    char other_ein[10];
    char mrn[12];
    char end_mrn[12];
    char seq_no[SEQ_SIZE+1];

    char name[48];
    char street_add[41];
    char city[26];
};

```



```

char state[11];
char zip_code[6];
long platter_slide;
long num_recs;
long final_offset;
long cum_offset;
};

```

```

/* EAMATE W2 Employee Information Detail Record
*/

```

```

/* This is the Complete structure of the Employee Information record
*/

```

```

struct EAMATE_W2EMPL_DETAIL
{

```

```

    char mrn[12];
    char ssn[12];
    char name[NAMELEN];
    char pens_ind[2];
    char defcomp_ind[2];
    char wages[9];
    char tips[9];
    char other[11];
    char fed_tax[11];
    char fica_tax[8];
    char adv_earn_inc[9];
    char med_wages[10];
    char med_tax[8];
    char ctrl_no[8];

    char street_add[28];
    char dep_care[9];
    char alloc_tips[9];
    char grp_insur[9];
    char uncoll_fica_tax[9];

```

```

    char city[19];
    char state[3];
    char zip_code[6];
    char defcomp[11];
    char sta[2];
    char fr_ben[11];
    char nqsec[11];
    char nqnot[11];
};

```

```

/* EAMATE W2 Employee Information Browse Record
*/

```

```

/* This is the Fields of the Employee Information records that will be
   * browsed upon. These records will make up the browse file.
   */

```

```

struct EAMATE_W2EMPL_BRW
{

```

```

    char ssn[12];
    char name[NAMELEN];
    char wages[9];
    char tips[9];
    char fica_tax[8];
    char other[11];
    char mrn[12];

```

```

char seq_no[SEQ_SIZE+1];
char wage_type[2];
long record_loc;
};

```

```

/* EAMATE W2 Intermediate Total Record
*/

```

```

/* This is the Complete structure of the Intermediate Total record
*/

```

```

struct EAMATE_W2INTERMED_TOT
{

```

```

    char proc_wages[12];
    char rep_wages[12];
    char proc_tips[12];
    char rep_tips[12];
    char proc_other[12];
    char rep_other[12];
    char proc_fed_tax[12];
    char rep_fed_tax[12];
    char proc_fica_tax[12];
    char rep_fica_tax[12];
    char proc_earn_inc[13];
    char rep_earn_inc[13];
    char proc_defcomp[12];
    char rep_defcomp[12];
    char proc_nonqual[12];
    char rep_nonqual[12];
    char ctrl_no[8];

```

```

    char proc_med_wages[13];
    char rep_med_wages[13];
    char proc_med_tax[12];
    char rep_med_tax[12];
};

```

```

/* EAMATE W2 Final Total Record
*/

```

```

/* This is the Complete structure of the Final Total record
*/

```

```

struct EAMATE_W2FINAL_TOT
{

```

```

    char proc_wages[15];
    char rep_wages[15];
    char proc_tips[14];
    char rep_tips[14];
    char proc_other[15];
    char rep_other[15];
    char proc_fed_tax[14];
    char rep_fed_tax[14];
    char proc_fica_tax[14];
    char rep_fica_tax[14];
    char proc_earn_inc[14];
    char rep_earn_inc[14];
    char proc_items[8];
    char rep_items[8];

    char proc_defcomp[15];
    char rep_defcomp[15];

```

```
char proc_nonqual[15];
char rep_nonqual[15];
char proc_med_wages[15];
char rep_med_wages[15];
char proc_med_tax[15];
char rep_med_tax[15];
};
```

```
/* EAMATE_W2 Cumulative EIN Total Record
```

```
 *
 * This is the Complete structure of the Cumulative EIN Total record
 */
```

```
struct EAMATE_W2CUMEIN_TOT
{
    char proc_wages[15];
    char proc_tips[14];
    char proc_other[15];
    char proc_fed_tax[14];
    char proc_fica_tax[14];
    char proc_earn_inc[14];
    char proc_items[8];
};
```





```

/*
 * params.h
 * version 3
 * 09/08/93
 *
 * by Natalie Willman
 *
 * This header file defines the parameters used to configure the
 * method of searching and indexing of the data files, as well as
 * some general structure definitions and function prototypes in
 * the general.c module.
 */

/* Define Statements */
#define GRAM_SIZE 2 /* Gram size to be used in parsing name */
#define SEQ_SIZE 3 /* Sequence length for numbering reports */
#define MAXGRAMS 26*26 /* Maximum number of gram combinations */
#define MAX_GRAM_SIZE 4 /* Maximum gram size allowed for ssn parse */
#define NAMELEN 28 /* Length of the index key before parse */
#define KEYLEN MAX_GRAM_SIZE+1 /* maximum length of index key */
#define ARRAY_SIZE NAMELEN-GRAM_SIZE+1 /* number grams after parse */
#define TRUE 1
#define FALSE 0
#define TRUE 1 /* Boolean Value for true */
#define FALSE 0 /* Boolean Value for false */

#define SUCCESS 1 /* Boolean Value for Successful return */
#define ERROR -1 /* Boolean Value for Error Return */
#define OVERLAP TRUE /* boolean value for gram parse type */
#define DUPEFILE TRUE /* boolean value for duplicate rec handling */
#define MAXDUPE 512 /* MAX duplicate recs to be held in memory */
#define IDF FALSE /* boolean value for weight (IDF) calc */
#define DICE TRUE /* boolean value for weight (DICE) calc */
#define PRUNE TRUE /* boolean value for pruning based upon DICE */
#define PRUNE_WEIGHT 33 /* Minimum DICE weight not pruned */
#define FPRUNE TRUE /* boolean value-prune based upon freq ratio */
#define FPRUNE_LEVEL 0.50 /* Minimum frequency ratio that is pruned */
#define MINGRAMS 5 /* minimum length of input key to be pruned */
#define DICE_GRAIN 150 /* range of weight values for DICE */
#define DICE_SCALE 100 /* scaling factor for calculating DICE wgt */
#define INITIALSCALE 1000 /* factor to determine threshold of initials */
#define INITIALFACTOR 2 /* factor to determine threshold of initials */
#define INITIALONLY 1.2 /* factor to determine threshold of initials */
#define FILENAME 50 /* length of filenames */
#define BLANKETSIZE 30 /* number of records in a blanket query */
#define BLANKETFACTOR 10 /* # of records from each section of report */
#define REALLOCFACTOR 10 /* Reallocation of memory factor */
#define BROWSENUM 1000 /* Number of recs for use in browse request */
#define COMMANDLENGTH 100 /* length of string for "system" commands */
#define KEEPSTATS FALSE /* stats of gram count indicator */
#define TESTRUN TRUE /* is this a evaluation run or a real run */
#define SET_SIZE 50 /* # of matches to convert during search */

/* Record Identifier Byte
 * This will be the first byte of each record, identifying the
 * type of record
 */
#define MATE_W2EH 0 /* W2 Employee Header */
#define MATE_W2EI 1 /* W2 Employee Information */
#define MATE_W2IT 2 /* W2 Intermediate Total */
#define MATE_W2FT 3 /* W2 Final Total */

/* Define MATE_W2CE 4 /* W2 Cumulative EIN */
/* Define MATE_W2CEI 5 /* W2C Employee Header */
/* Define MATE_W2CEI 6 /* W2C Employee Information */
/* Define MATE_W2CFT 7 /* W2C Final Total */
/* Define MATE_W2CEH 8 /* W2 Employee Header */
/* Define MATE_W2CEI 9 /* W2 Employee Information */
/* Define MATE_W2IT 10 /* W2 Intermediate Total */
/* Define MATE_W2FT 11 /* W2 Final Total */
/* Define MATE_W2CEI 12 /* W2 Cumulative EIN */
/* Define MATE_W2CEH 13 /* W2C Employee Header */
/* Define MATE_W2CEI 14 /* W2C Employee Information */
/* Define MATE_W2CFT 15 /* W2C Final Total */

/* Function Prototypes */
void cr_browse_filename();
void error_exit();
int parse_ssn();
int parse_name();

/*
 * structure of User's Query
 * received from the User interface
 */

struct USER_QUERY
{
    char year[5];
    char ein[11];
    char est[5];
    char seq_no[SEQ_SIZE+1];
    char first[13];
    char last[16];
    char ssn[12];
    char offset[30];
};

/*
 * structure of the inverted file
 * for the index of employers
 */

struct EMPLR_IDX
{
    char ein[11];
    long offset;
};

/* stats structure */
struct STATS
{
    long count;
    long numgrams[MAXGRAMS];
};

```

```
/* parse control file structure */
```

```
struct CTRL_FILE
```

```
{
    char ein[11];
    char seq[SEQ_SIZE+1];
    char browse_loc[3];
};
```

```
/* structure of the test data kept by search */
```

```
struct TEST_DATA
```

```
{
    short user;
    char year[5];
    char ein[11];
    char first[13];
    char last[16];
    char searchname[30];
    char ssn[12];
    long numgrams;
    long nummatch;
    long recspruned;
    long grprune;
    long etime;
    long wtime;
    long mem;
    double ctime;
};
```





```
/*
 * paramsemp1r.h
 * version 3
 * 9/8/93
 *
 * by Natalie Willman
 *
 * This header file defines the parameters used to configure the
 * method of searching and indexing of the data files, as well as
 * some general structure definitions and function prototypes in
 * the general.c module. Used for indexsemp1r.c
 */

/* Define Statements */
#define NAMELEN 28 /* not used except for in eamteststruct.h */
#define KEYLEN 11 /* maximum length of index key */
#define TRUE 1 /* Boolean Value for True */
#define FALSE 0 /* Boolean Value for False */
#define DUPEFILE FALSE /* boolean value for duplicate rec handling */
#define MAXDUPE 1024 /* MAX duplicate recs to be held in memory */
#define SEQ_SIZE 3

/*
 * structure of the inverted file
 * for the index of employers
 */

struct EMPLR_IDX
{
    char ein[11];
    long offset;
};
```

# Makefile

Tue Jan 4 09:41:20 1994

1

```
#
# Bottom Level Makefile (~/.ssapilot/src/lib/btree_data)
#
# by Natalie Willman
#
# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependencies for the source files.
#
# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../../..
LIBDIR       = $(PROJECT_ROOT)/lib
BINDIR       = $(PROJECT_ROOT)/bin
INCDIR       = $(PROJECT_ROOT)/include

# this is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE   = libbtree_data.a
SRC           = btree_data.c
LIBS          =
CLIBS         = -lm
OBJ           = btree_data.o
CFLAGS        = -I$(INCDIR) -L$(LIBDIR)
CC            = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(LIBDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(LIBDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependancies for each of the source files
depend : $(SRC)
    $(CC) -M $(CFLAGS) $(SRC) > dependlist
    sed -e '1,/^# DO NOT DELETE/id' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv Makefile Makefile.bak
    mv mm.tmp Makefile
```

```
rm -f dependlist

# directive for the executable
$(EXECUTABLE) : $(OBJ)
    ar rv $@ $(OBJ)

# directive for the executable in the binary directory
$(LIBDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(LIBDIR)

.o.a :
    ar rv $@ $(OBJ)

.c.o :
    $(CC) -c $(CFLAGS) $<

.c.a :
    $(CC) -c $(CFLAGS) $<
    ar rv $@ $(OBJ)

# DO NOT DELETE THIS LINE - make depend uses it
btree_data.o: btree_data.c
btree_data.o: /usr/include/stdio.h
btree_data.o: /usr/include/string.h
btree_data.o: ../../include/params.h
btree_data.o: ../../include/btreestruct.h
```

```

/*
 * btree.c
 * version 3.0
 * 09/08/93
 *
 * by Natalie Willman
 * NIST
 *
 * This module contains functions specific to creating, adding records, and
 * searching a btree. Additional modules are needed to provide record
 * specific functions (eamate.c), general functions (general.c)
 * and search/index main control modules specific to file being indexed
 * (search.c/index.c/indexemplr.c).
 *
 * List of Functions:
 *
 * search_tree() : starts btree search for match
 * find_record() : recursively searches tree for a key match
 * btree_create() : creates the B+ tree root
 * btree_walk() : walk a memory tree and print it to the file
 * btree_insert() : inserts an index key into the B+ tree
 * insert_nonfull() : inserts an index key into a nonfull btree node
 * split_child() : splits a full btree node into two nodes
 * allocate_node() : allocates a node of type NODE struct in memory
 * allocate_dupe() : allocates a dupe structure of type KEYLIST
 * disk_update() : updates a node in the disk index file
 * disk_read() : reads a node from the disk index file
 * binsearch() : binary search on an array (for keys in a node)
 * add_dupe() : adds a duplicate key to the btree
 * add_new() : adds a new key to the btree
 */

/* Include files */
#include <stdio.h>
#include <string.h>

#include "params.h"
#include "btreestruct.h"

/*
 * search_tree()
 *
 * This function searches a btree for the key matching the
 * specified data, and will return the file offset of the data, and
 * the frequency of the key within the file.
 *
 * Input: Character array containing the key for which to be searched, "key"
 * File pointer to the B+ tree file, "btree_fil"
 * Frequency of the key being searched (returned), "frequency"
 *
 * Output: File offset of the record in the data file
 * Frequency value of the key, "frequency", is filled
 */

long search_tree(key, btree_fil, frequency)
char key[];
FILE *btree_fil;
long *frequency;
{
    struct NODE *root, rootnode; /* Current root in search (sub)tree */
    long location; /* File offset location of record */

    long search_tree(key, btree_fil, frequency)
    char key[];
    FILE *btree_fil;
    long *frequency;
    {
        struct NODE *root, rootnode; /* Current root in search (sub)tree */
        long location; /* File offset location of record */

        /* Initialize variables and open index file */
        root = &rootnode;

        /* Read the root node, and call search routine */
        disk_read(root, 0L, btree_fil);
        location = find_record(root, key, btree_fil, frequency);

        /* Return location of the record, -1 if not found */
        return(location);
    }

    /* find_record()
    *
    * This function will accept as input the current root of the
    * search (sub)tree and the key to search for. It will recursively
    * traverse the tree until a leaf node is reached. If the leaf
    * node contains a match, the file offset of the match is returned,
    * otherwise a -1 is returned
    *
    * Input: Character array containing the key to be searched upon, "key"
    * pointer to a structure containing root of search (sub)tree, "node"
    * Frequency of the key being searched (returned), "frequency"
    * File containing the B+ tree, "btree_fil"
    *
    * Output: File offset of the record in the data file
    * Frequency value of the key is filled
    */
    long find_record(node, key, btree_fil, frequency)
    struct NODE *node;
    char key[];
    FILE *btree_fil;
    long *frequency;
    {
        int i; /* Counter Variable */

        /* do a binary search on the keys in the node until a key is
        * reached that is greater than or equal to the search key
        */
        i = binsearch(key, node->key, node->count);

        /* If the node is a leaf, then check to see if the search key is
        * equal to the node key. If it is, return the offset of that
        * key, otherwise return -1
        */
        if (node->leaf == TRUE)
        {
            if ((i < node->count) && (strcmp(key, node->key[i]) == 0))
            {
                *frequency = node->freq[i];
                return (node->branch[i]);
            }
            else
            {
                return (-1);
            }
        }
        /* If the node is not a leaf, then read in the child that leads in
        * the direction of the match, and recursively continue the tree
        */
        /* search
        else
        {
            disk_read(node, node->branch[i], btree_fil);
            find_record(node, key, btree_fil, frequency);
        }
    }

```



```

    }
}

/*
 * btree_create()
 *
 * This function will create the root node in the b+ tree. The
 * root node is stored in memory and at file offset location 0
 * in the b+ tree file.
 *
 * Input: location of current write offset in tree, "btree_loc"
 *
 * Output: Pointer to the root node of the b+ tree (in memory)
 */
struct MEMNODE * btree_create(btree_loc)
long *btree_loc;
{
    struct MEMNODE *root; /* pointer to the root node of the b+ tree */

    /* Allocate a space in memory for the root node, and initialize it
     * to having no keys, being a leaf node, and being the end of the
     * linked list of leaf nodes
     */
    root = allocate_node(btree_loc);
    root->count = 0;
    root->leaf = TRUE;
    root->branch.mem[MAXCHILD] = NULL;

    /* return the memory pointer to the root
     */
    return(root);
}

/*
 * btree_walk
 *
 * This function traverses a b+ tree and prints it to the file.
 *
 * Input: pointer to the root of the tree, "root"
 *        file to which to write the tree, "btree_fil"
 *
 * Output: tree with root "root" is written to "btree_fil"
 */
btree_walk(root, btree_fil)
FILE *btree_fil;
struct MEMNODE *root;
{
    int i;
    struct MEMNODE *temp;

    /* If the node is not a leaf ...
     */
    if (root->leaf != TRUE)
    {
        /* recursively call this function for each of this nodes children */
        for (i = 0; i <= root->count; i++)
            btree_walk(root->branch.mem[i], btree_fil);
        /* for each of the children, set the pointer to it in the node to
         * the offset value in the self offset field of the child node
         */
        /* must be done post-call to btree_walk, because this destroys the
         * memory pointer. free the child nodes
         */
    }
}

for (i = 0; i <= root->count; i++)
{
    temp = root->branch.mem[i];
    root->branch.disk[i] = (root->branch.mem[i])->self_offset;
    free(temp);
}

/* write the node information to disk
 */
disk_update(root, btree_fil);
}
else
{
    /* If not at the end of the leaf chain, then set the pointer to
     * the next leaf to the self offset value of the node pointed to
     * by the pointer in the MAXCHILD location.
     */
    if (root->branch.mem[MAXCHILD] != NULL)
        root->branch.disk[MAXCHILD] = (root->branch.mem[MAXCHILD])->self_offset;
    /* otherwise, set the pointer to the next leaf to a -1, signalling
     * end of leaf list
     */
    else
        root->branch.disk[MAXCHILD] = -1;
    /* write the node information to disk
     */
    disk_update(root, btree_fil);
}
}

/*
 * btree_insert()
 *
 * This function will take as input the record to be inserted,
 * check to see if the root of the tree is full (if so, split the
 * root, thereby increasing the height of the tree), and insert
 * the record into the tree.
 *
 * Input: Pointer to a pointer to root of tree (to allow mods), "root"
 *        Pointer to the array containing the key, "key"
 *        Integer value of the location of the record, "record_loc"
 *        pointer to duplicate file, "dupe_fil"
 *        pointer to next write location in btree file, "btree_loc"
 *        Weight of the key being inserted (length in grams), "weight"
 *
 * Output: The record is inserted into a non-full node of the tree,
 *        and, if the root is full, it is split.
 */
btree_insert(root, key, record_loc, dupe_fil, btree_loc, weight)
struct MEMNODE **root;
char key[];
FILE *dupe_fil;
long *btree_loc;
long record_loc;
int weight;
{
    struct MEMNODE *temp;

    /* If the root node is full, split the node
     */
    if ((*root)->count == MAXCHILD)
    {
        /* assign root to temp value - necessary if node is split
         */
        temp = *root;
        /* allocate a new node for the root
         */
        *root = allocate_node(btree_loc);
    }
}

```

```

/* set new root node's disk offset to 0, and the new nodes offset to
/* the value assigned by allocate_node() - root must be at 0 for
/* tree traversal in the search routine
temp->self_offset = (*root)->self_offset;
(*root)->self_offset = 0;

/* Initialize the new root, and link it to its child, the previous
/* root node
(*root)->leaf = FALSE;
(*root)->count = 0;
(*root)->branch.mem[0] = temp;

/* split the child node, and free the space allocated to the old root */
split_child(*root, 0, temp, btree_loc);
}

/* Insert the record into the btree
insert_nonfull(*root, key, record_loc, dupe_fil, btree_loc, weight);
}

/* Insert_nonfull()
/* This function will take as input the record to be inserted,
/* the node in which it should be inserted, and the location of
/* the record. The data will be inserted into the node.
/* Input: pointer to the a node in the tree (for first invocation, it is
/* the root), "node"
/* pointer to the array containing the key, "key"
/* long integer containing the location of the record, "record_loc"
/* pointer to duplicate file, "dupe_fil"
/* Next write location in btree file, "btree_loc"
/* Weight of the record (length in grams), "weight"
/* Output: the key is inserted into a node
*/

insert_nonfull(node, key, record_loc, dupe_fil, btree_loc, weight)
struct MEMNODE *node;
char key[];
FILE *dupe_fil;
long record_loc;
long *btree_loc;
int weight;
{
    int i; /* counter variable */
    struct MEMNODE *child, cnode; /* child node variable */
    /* Initialize variables */
    child = &cnode;

    /* If the current node is a leaf, the key can be inserted into
    /* this node (either the root is a leaf, or the tree has been
    /* traversed to reach the appropriate leaf for insertion)
    if (node->leaf == TRUE)
    {
        if (DUPEFILE == FALSE) /* If dupes are to be put in the tree
        {
            /* Starting at the last (greatest) key in the leaf, work
            /* go to the appropriate child, and check to see if the
            /* do a binary search on the keys in the node until a key
            /* is reached that is greater than or equal to the search
            /* key. This is the child to process next
            i = binsearch(key, node->key, node->count);
            }
        }
        else /* If the node is not a leaf */
        {
            /* copy the index key into the location at "i", and store the
            /* record location of the record in the data file, increment
            /* the node counter, reset number of dupes, and the dupe offset
            strcpy(node->key[i], key);
            node->branch.dup[i] = add_new(record_loc, weight);
            node->num_dupe[i] = 1;
            node->dupe_offset[i] = -1;
            node->count++;
        }
    }
    else /* If the key exists, write to dupe file
    if (strcmp(key, node->key[i]) == 0)
        add_dupe(record_loc, i, dupe_fil, node->branch.dup,
            node->num_dupe, node->dupe_offset, weight);
    else /* If key does not exist, add to tree and write to dupe
    {
        /* Starting at the last (greatest) key in the leaf, work
        /* backwards down the leaf list, moving the keys up an index
        /* in the node, until the next key is less than the value
        /* of the input key. "i" now points to the position to
        /* insert the index key.
        i = node->count;
        while( (i > 0) && (strcmp(key, node->key[i-1]) < 0) )
        {
            strcpy(node->key[i], node->key[i-1]);
            node->branch.dup[i] = node->branch.dup[i-1];
            node->num_dupe[i] = node->num_dupe[i-1];
            node->dupe_offset[i] = node->dupe_offset[i-1];
            i--;
        }
        /* copy the index key into the location at "i", and store the
        /* record location of the record in the data file, increment
        /* the node counter, reset number of dupes, and the dupe offset
        strcpy(node->key[i], key);
        node->branch.dup[i] = add_new(record_loc, weight);
        node->num_dupe[i] = 1;
        node->dupe_offset[i] = -1;
        node->count++;
    }
}

/* search for correct location in the node to insert
i = binsearch(key, node->key, node->count);

/* If the key exists, write to dupe file
if (strcmp(key, node->key[i]) == 0)
    add_dupe(record_loc, i, dupe_fil, node->branch.dup,
        node->num_dupe, node->dupe_offset, weight);
else /* If key does not exist, add to tree and write to dupe
{
    /* Starting at the last (greatest) key in the leaf, work
    /* backwards down the leaf list, moving the keys up an index
    /* in the node, until the next key is less than the value
    /* of the input key. "i" now points to the position to
    /* insert the index key.
    i = node->count;
    while( (i > 0) && (strcmp(key, node->key[i-1]) < 0) )
    {
        strcpy(node->key[i], node->key[i-1]);
        node->branch.dup[i] = node->branch.dup[i-1];
        node->num_dupe[i] = node->num_dupe[i-1];
        node->dupe_offset[i] = node->dupe_offset[i-1];
        i--;
    }
    /* copy the index key into the location at "i", and store the
    /* record location of the record in the data file, increment
    /* the node counter, reset number of dupes, and the dupe offset
    strcpy(node->key[i], key);
    node->branch.dup[i] = add_new(record_loc, weight);
    node->num_dupe[i] = 1;
    node->dupe_offset[i] = -1;
    node->count++;
}

/* do a binary search on the keys in the node until a key
/* is reached that is greater than or equal to the search
/* key. This is the child to process next
i = binsearch(key, node->key, node->count);
}

/* go to the appropriate child, and check to see if the

```

```

/* node is full (does not allow the recursion to descend to
/* a full child)
child = node->branch.mem[l];
if (child->count == MAXCHILD)
{
    /* If the node is full, split the child
    split_child(node, l, child, btree_loc);
    /* Once the child is split, see whether the input key is
    /* greater than the new key inserted in the parent node
    /* If so, increment l, and read in that child
    if (strcmp(key, node->key[l]) > 0)
    {
        l++;
        child = node->branch.mem[l];
    }
}

/* recursively call insert_nonfull() to insert a key. Stops
/* recursion when a leaf node is reached
insert_nonfull(child, key, record_loc, dupe_fil, btree_loc, weight);
}

/*
* split_child()
*
* This function takes as input a node that is full, its parent node
* and an index to where the child is located within the parent, and
* splits the child node into two non-full nodes
*
* Input: pointer to the parent node, "parent"
*         location of the child within the parent node, "index"
*         pointer to the child node, "child"
*         next write location in btree file, "btree_loc"
*
* Output: the node, "child", is split, and parent is expanded to
*         include a new child key.
*/

split_child(parent, index, child, btree_loc)
struct MEMNODE *parent, *child;
long *btree_loc;
int index;
{
    struct MEMNODE *newchild;
    int l;

    /* allocate a node for the new child and initialize its leaf
    /* and count fields
    newchild = allocate_node(btree_loc);
    newchild->leaf = child->leaf;
    newchild->count = MINCHILD - 1;

    /* Copy the higher keys from the old child to the new child
    for (l = 0; l < MINCHILD - 1; l++)
        strcpy(newchild->key[l], child->key[l+MINCHILD]);

    /* copy the higher branch pointers from the old child to the
    /* new child. In internal nodes, these are pointers to its
    /* children, in leaf nodes, these are pointers to records in
    /* the data file
    if (newchild->leaf == FALSE)
        for (l = 0; l < MINCHILD; l++)
            newchild->branch.mem[l] = child->branch.mem[l+MINCHILD];

    else
    {
        if (DUPEFILE == FALSE)
        {
            for (l = 0; l < MINCHILD-1; l++)
                newchild->branch.disk[l] = child->branch.disk[l+MINCHILD];
        }
        else
        {
            for (l = 0; l < MINCHILD-1; l++)
            {
                newchild->branch.dup[l] = child->branch.dup[l+MINCHILD];
                newchild->num_dupe[l] = child->num_dupe[l+MINCHILD];
                newchild->dupe_offset[l] = child->dupe_offset[l+MINCHILD];
            }
        }
    }

    /* If we are at leaf level, copy the pointer to the next leaf
    /* in the linked list from the old child to the new child.
    /* This pointer is stored in the last branch array value, and
    /* is copied because this new leaf will be inserted after the
    /* current leaf in the chain
    if (newchild->leaf == TRUE)
        newchild->branch.mem[MAXCHILD] = child->branch.mem[MAXCHILD];

    /* If the current child is not a leaf, set the value of its
    /* node count to MINCHILD-1. If it is a leaf, set the count value
    /* to MINCHILD and its pointer to the next leaf to the new child.
    /* If it is a leaf being split, it is important to set the
    /* count value to MINCHILD because it is a COPY of the record that
    /* is going to the parent, and we do not want to lose the
    /* record. However, if it is not a leaf, the index key does
    /* not need to be retained in that node (otherwise, you get
    /* two nodes pointing to the same child)
    if (child->leaf == FALSE)
        child->count = MINCHILD-1;
    else
    {
        child->count = MINCHILD;
        child->branch.mem[MAXCHILD] = newchild;
    }

    /* Move all of the keys (and corresponding branches) greater than
    /* the index key in the parent node up one array slot in the node.
    /* Link the new child node into the parent node at the index loc.
    for (i = parent->count; i > index; i--)
        parent->branch.mem[i+1] = parent->branch.mem[i];
    parent->branch.mem[index+1] = newchild;
    for (i = parent->count; i > index; i--)
        strcpy(parent->key[i], parent->key[i-1]);
    strcpy(parent->key[index], child->key[MINCHILD-1]);

    /* Increment the parent count
    parent->count++;
}

/* allocate_node()
*
* This function allocates memory for a node, and returns a pointer
* to that memory location, and updates the location that this node
* will be written to the tree file
*
* Input: pointer to the next write location in the btree file, "btree_loc"

```



```

/*
 * Output: memory for a new node is allocated, and a pointer to
 * that memory is returned
 */
struct MEMNODE *allocate_node(btree_loc)
long *btree_loc;
{
    struct MEMNODE *node;
    int i;

    node = (struct MEMNODE *) malloc(sizeof(struct MEMNODE));
    if (node == NULL)
        error_exit("ERROR: Unable to allocate memory");
    node->self_offset = *btree_loc;
    for (i = 0; i < MAXCHILD; i++)
        node->dupe_offset[i] = -1;
    *btree_loc = *btree_loc + sizeof(struct NODE);
    return (node);
}

/*
 * allocate_dupe()
 *
 * This function allocates memory for a linked list structure, and
 * returns a pointer to that memory location
 *
 * Input: none
 *
 * Output: a struct of type KEYLIST is allocated, and a pointer to
 * that memory is returned
 */
struct KEYLIST *allocate_dupe()
{
    struct KEYLIST *dupe;

    dupe = (struct KEYLIST *) malloc(sizeof(struct KEYLIST));
    if (dupe == NULL)
        error_exit("ERROR: Unable to allocate memory");
    return (dupe);
}

/*
 * disk_update()
 *
 * This function updates a node's information that has already been written
 * to the disk file
 *
 * Input: Pointer to the node data, "node"
 *         pointer to the file to hold the b+ tree, "btree_fil"
 *
 * Output: The index file data for this node is updated
 */
disk_update(node, btree_fil)
struct MEMNODE *node;
FILE *btree_fil;
{
    int num;

    struct NODE disknode;

    /* copy memory node to disk node
     * disknode.count = node->count;
     * disknode.leaf = node->leaf;
     * for (num = 0; num < MAXCHILD; num++)
     * {
     *     strcpy(disknode.key[num], node->key[num]);
     *     disknode.branch[num] = node->branch.disk[num];
     *     disknode.freq[num] = node->num_dupe[num];
     * }
     * disknode.branch[MAXCHILD] = node->branch.disk[MAXCHILD];

     * seek to the offset of this node, and write the node to the file */
    fseek(btree_fil, node->self_offset, 0);
    if (! fwrite(&disknode, sizeof(struct NODE), 1, btree_fil) == 0)
        return (FALSE);
    else
        return (TRUE);
}

/*
 * disk_read()
 *
 * This function reads a node value from the disk file
 *
 * Input: Pointer to the node to store the data, "node"
 *         Offset to the node to read, "offset"
 *         File pointer to the btree file, "btree_fil"
 *
 * Output: The data in the node at "offset" in the btree file is
 *         copied to "node"
 */
disk_read(node, offset, btree_fil)
struct NODE *node;
long offset;
FILE *btree_fil;
{
    /* seek to the specified offset, and read the node data */
    fseek(btree_fil, offset, 0);
    if (! fread(node, sizeof(struct NODE), 1, btree_fil) == 0)
        return (FALSE);
    else
        return (TRUE);
}

/*
 * btree_search()
 *
 * This function will take an input key, and an array of other keys
 * and do a binary search on the array, passing back the index to the
 * key that is closest to but greater than the input key (or 1+ that
 * number of all of the keys are less than the input key.
 *
 * Input: Input key to search upon, "key"
 *         array to search within, "searchspace"
 *         length of search array, "count"
 *
 * Output: an integer value corresponding to the index to the key
 *         in the search array that is closest to but greater than the
 *         input key
 */

```

```

/*
Int binsearch(key, searchspace, count)
char key[], searchspace[MAXCHILD][KEYLEN];
int count;
{
    int left, right, index;

    /* Set up left and right bounds based upon the number of children */
    /* a node can have (1 + count)
    left = 0;
    right = count;

    /* While there are records between the left and right boundary.. */
    while (right > left)
    {
        /* Determine midpoint
        index = (left + right)/2;

        /* If the input key is greater than the midpoint, then change
        /* left and right bounds to only search upper half of data
        if (strcmp(key, searchspace[index]) > 0)
            left = index+1;
        /* Otherwise, change the bounds to only search the lower half
        /* of the data
        else
            right = index;
    }

    /* Once the left and right boundaries are the same, return the
    /* array index
    return(left);
}

/*
* add_dupe
*
* this function adds a duplicate record to the dupe file for a
* given key
*
* Input:  record location of the record for the key being added, "record_loc"
*         index to key location in the btree node, "i"
*         pointer to the duplicate file, "dupe_fil"
*         pointer to the start of the linked lists in this node, "branch"
*         number of duplicate offsets currently in each of the dupe lists, "num_dupe"
*         offset link to the linked lists in the file for each linked list, "offset_list"
*
*         weight of this record in grams, "weight"
*
* Output: this record is added to the dup list for the key
*
*
*
add_dupe(record_loc, i, dupe_fil, branch, num_dupe, offset_list, weight)
long record_loc; /* record location of the record */
int i, weight;
FILE *dupe_fil; /* node branches
struct KEYLIST *branch[];
long num_dupe[];
long offset_list[];
{
    struct KEYLIST *dupekey; /* linked list structure for dupe file
    struct DUPELLIST dupelist;

```

```

# Bottom Level Makefile (~/.ssapilot/src/lib/btree_emplr)
# by Natalie Willman

# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependancies for the source files.

# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../../..
LIBDIR       = $(PROJECT_ROOT)/lib
BINDIR       = $(PROJECT_ROOT)/bin
INCDIR       = $(PROJECT_ROOT)/include

# this is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE   = libtree_emplr.a
SRC          = btree_emplr.c
LIBS         =
CLIBS        = -lm
OBJ          = btree_emplr.o
CFLAGS       = -I$(INCDIR) -L$(LIBDIR)
CC           = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(LIBDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(LIBDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependancies for each of the source files
depend : $(SRC)
    $(CC) -M $(CFLAGS) $(SRC) > dependlist
    sed -e '1,/^\# DO NOT DELETE/id' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv Makefile Makefile.bak
    mv mm.tmp Makefile

```

```

rm -f dependlist

# directive for the executable
$(EXECUTABLE) : $(OBJ)
    ar rv $@ $(OBJ)

# directive for the executable in the binary directory
$(LIBDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(LIBDIR)

.o.a :
    ar rv $@ $(OBJ)

.c.o :
    $(CC) -c $(CFLAGS) $<

.c.a :
    $(CC) -c $(CFLAGS) $<
    ar rv $@ $(OBJ)

# DO NOT DELETE THIS LINE -- make depend uses it
btree_emplr.o: btree_emplr.c
btree_emplr.o: /usr/include/stdio.h
btree_emplr.o: /usr/include/string.h
btree_emplr.o: ../../include/paramsemplr.h
btree_emplr.o: ../../include/btreestruct.h

```











```

/* node is full (does not allow the recursion to descend to
/* a full child)
child = node->branch.mem[i];
if (child->count == MAXCHILD)
{
    /* If the node is full, split the child
    split_child(node, i, child, btree_loc);
    /* Once the child is split, see whether the input key is
    /* greater than the new key inserted in the parent node
    /* If so, increment i, and read in that child
    if (strcmp(key, node->key[i]) > 0)
    {
        i++;
        child = node->branch.mem[i];
    }
}

/* recursively call insert_nonfull() to insert a key. Stops
/* recursion when a leaf node is reached
insert_nonfull(child, key, record_loc, dupe_fil, btree_loc, weight);
}

/* split_child()
* This function takes as input a node that is full, its parent node
* and an index to where the child is located within the parent, and
* splits the child node into two non-full nodes
*
* Input: pointer to the parent node, "parent"
* location of the child within the parent node, "index"
* pointer to the child node, "child"
* next write location in btree file, "btree_loc"
*
* Output: the node, "child", is split, and parent is expanded to
* include a new child key.
*/

split_child(parent, index, child, btree_loc)
struct MEMNODE *parent, *child;
long *btree_loc;
int index;
{
    struct MEMNODE *newchild;
    int i;

    /* allocate a node for the new child and initialize its leaf */
    /* and count fields
    newchild = allocate_node(btree_loc);
    newchild->leaf = child->leaf;
    newchild->count = MINCHILD - 1;

    /* Copy the higher keys from the old child to the new child */
    for (i = 0; i < MINCHILD - 1; i++)
        strcpy(newchild->key[i], child->key[i+MINCHILD]);

    /* copy the higher branch pointers from the old child to the
    /* new child. In internal nodes, these are pointers to its
    /* children, in leaf nodes, these are pointers to records in
    /* the data file
    if (newchild->leaf == FALSE)
        for (i = 0; i < MINCHILD; i++)
            newchild->branch.mem[i] = child->branch.mem[i+MINCHILD];
}

else
    if (DUPEFILE == FALSE)
    {
        for (i = 0; i < MINCHILD-1; i++)
            newchild->branch.disk[i] = child->branch.disk[i+MINCHILD];
    }
    else
    {
        for (i = 0; i < MINCHILD-1; i++)
        {
            newchild->branch.dup[i] = child->branch.dup[i+MINCHILD];
            newchild->num_dupe[i] = child->num_dupe[i+MINCHILD];
            newchild->dupe_offset[i] = child->dupe_offset[i+MINCHILD];
        }
    }

    /* If we are at leaf level, copy the pointer to the next leaf */
    /* in the linked list from the old child to the new child. */
    /* This pointer is stored in the last branch array value, and */
    /* is copied because this new leaf will be inserted after the */
    /* current leaf in the chain
    if (newchild->leaf == TRUE)
        newchild->branch.mem[MAXCHILD] = child->branch.mem[MAXCHILD];

    /* If the current child is not a leaf, set the value of its
    /* node count to MINCHILD-1. If it is a leaf, set the count value */
    /* to MINCHILD and its pointer to the next leaf to the new child. */
    /* If it is a leaf being split, it is important to set the
    /* count value to MINCHILD because it is a copy of the record that
    /* is going to the parent, and we do not want to lose the
    /* record. However, if it is not a leaf, the index key does
    /* not need to be retained in that node (otherwise, you get
    /* two nodes pointing to the same child)
    if (child->leaf == FALSE)
        child->count = MINCHILD-1;
    else
    {
        child->count = MINCHILD;
        child->branch.mem[MAXCHILD] = newchild;
    }

    /* Move all of the keys (and corresponding branches) greater than
    /* the index key in the parent node up one array slot in the node.
    /* Link the new child node into the parent node at the index loc.
    for (i = parent->count; i > index; i--)
        parent->branch.mem[i+1] = parent->branch.mem[i];
    parent->branch.mem[index+1] = newchild;
    for (i = parent->count; i > index; i--)
        strcpy(parent->key[i], parent->key[i-1]);
    strcpy(parent->key[index], child->key[MINCHILD-1]);

    /* Increment the parent count
    parent->count++;
}

/* allocate_node()
* This function allocates memory for a node, and returns a pointer
* to that memory location, and updates the location that this node
* will be written to the tree file
*
* Input: pointer to the next write location in the btree file, "btree_loc"

```

```

/*
 * Output: memory for a new node is allocated, and a pointer to
 * that memory is returned
 */
struct MEMNODE *allocate_node(btree_loc)
long *btree_loc;
{
    struct MEMNODE *node;
    int i;

    node = (struct MEMNODE *) malloc(sizeof(struct MEMNODE));
    if (node == NULL)
        error_exit("ERROR: Unable to allocate memory");
    node->self_offset = *btree_loc;
    for (i = 0; i < MAXCHILD; i++)
        node->dupe_offset[i] = -1;
    *btree_loc = *btree_loc + sizeof(struct NODE);
    return (node);
}

/*
 * allocate_dupe()
 *
 * This function allocates memory for a linked list structure, and
 * returns a pointer to that memory location
 *
 * Input: none
 *
 * Output: a struct of type KEYLIST is allocated, and a pointer to
 * that memory is returned
 */
struct KEYLIST *allocate_dupe()
{
    struct KEYLIST *dupe;

    dupe = (struct KEYLIST *) malloc(sizeof(struct KEYLIST));
    if (dupe == NULL)
        error_exit("ERROR: Unable to allocate memory");
    return (dupe);
}

/*
 * disk_update()
 *
 * This function updates a node's information that has already been written
 * to the disk file
 *
 * Input: pointer to the node data, "node"
 * pointer to the file to hold the b+ tree, "btree_fil"
 *
 * Output: The index file data for this node is updated
 */
disk_update(node, btree_fil)
struct MEMNODE *node;
FILE *btree_fil;
{
    int num;

    struct NODE disknode;

    /* copy memory node to disk node
     * disknode.count = node->count;
     * disknode.leaf = node->leaf;
     * for (num = 0; num < MAXCHILD; num++)
     * {
     *     strcpy(disknode.key[num], node->key[num]);
     *     disknode.branch[num] = node->branch.disk[num];
     *     disknode.freq[num] = node->num_dupe[num];
     * }
     * disknode.branch[MAXCHILD] = node->branch.disk[MAXCHILD];

    /* seek to the offset of this node, and write the node to the file */
    fseek(btree_fil, node->self_offset, 0);
    if (fwrite(&disknode, sizeof(struct NODE), 1, btree_fil) == 0)
        return (FALSE);
    else
        return (TRUE);
}

/*
 * disk_read()
 *
 * This function reads a node value from the disk file
 *
 * Input: pointer to the node to store the data, "node"
 * Offset to the node to read, "offset"
 * File pointer to the btree file, "btree_fil"
 *
 * Output: The data in the node at "offset" in the btree file is
 * copied to "node"
 */
disk_read(node, offset, btree_fil)
struct NODE *node;
long offset;
FILE *btree_fil;
{
    /* seek to the specified offset, and read the node data */
    fseek(btree_fil, offset, 0);
    if (fread(node, sizeof(struct NODE), 1, btree_fil) == 0)
        return (FALSE);
    else
        return (TRUE);
}

/*
 * binsearch()
 *
 * This function will take an input key, and an array of other keys
 * and do a binary search on the array, passing back the index to the
 * key that is closest to but greater than the input key (or 1+ that
 * number of all of the keys are less than the input key.
 *
 * Input: input key to search upon, "key"
 * array to search within, "searchspace"
 * length of search array, "count"
 *
 * Output: an integer value corresponding to the index to the key
 * in the search array that is closest to but greater than the
 * input key
 */

```

```

/*
Int binsearch(key, searchspace, count)
char key[], searchspace[MAXCHILD][KEYLEN];
int count;
{
    int left, right, index;

    /* Set up left and right bounds based upon the number of children */
    /* a node can have (1 + count) */
    left = 0;
    right = count;

    /* While there are records between the left and right boundary... */
    while(right > left)
    {
        /* Determine midpoint
        index = (left + right)/2;

        /* If the input key is greater than the midpoint, then change
        /* left and right bounds to only search upper half of data
        if(strcmp(key, searchspace[index]) > 0)
            left = index+1;
        /* Otherwise, change the bounds to only search the lower half
        /* of the data
        else
            right = index;
    }

    /* Once the left and right boundaries are the same, return the
    /* array index
    return(left);
}

/*
* add_dupe
*
* this function adds a duplicate record to the dupe file for a
* given key
*
* Input: record location of the record for the key being added, "record_loc"
* index to key location in the btrees node, "i"
* pointer to the duplicate file, "dupe_fil"
* pointer to the start of the linked lists in this node, "branch"
* number of duplicate offsets currently in each of the dupe lists, "num_dupe"
* offset link to the linked lists in the file for each linked list, "offset_lis"
*
* weight of this record in grams, "weight"
*
* Output: this record is added to the dup list for the key
*
*
*
add_dupe(record_loc, i, dupe_fil, branch, num_dupe, offset_list, weight)
long record_loc; /* record location of the record
int i, weight;
FILE *dupe_fil;
struct KEYLIST *branch[]; /* node branches
long num_dupe[];
long offset_list[];
{
    struct KEYLIST *dupekey; /* linked list structure for dupe file
    struct DUPELIST dupelist;

```



```

#
# Bottom Level Makefile (~/.ssapilot/src/lib/general)
#
# by Natalie Willman
#
# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependancies for the source files.
#
# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../../..
LIBDIR      = $(PROJECT_ROOT)/lib
BINDIR      = $(PROJECT_ROOT)/bin
INCDIR      = $(PROJECT_ROOT)/include

# This is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE  = libgen_eamate.a
SRC          = eamate.c general.c
LIBS         =
CLIBS        = -lm
OBJ          = eamate.o general.o
CFLAGS       = -I$(INCDIR) -I$(LIBDIR)
CC           = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(LIBDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(LIBDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependancies for each of the source files
depend : $(SRC)
    $(CC) -M $(CFLAGS) $(SRC) > dependlist
    sed -e '1,/^# DO NOT DELETE/d' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv Makefile Makefile.bak
    mv mm.tmp Makefile

```

```

rm -f dependlist

# directive for the executable
$(EXECUTABLE) : $(OBJ)
    ar rv $@ $(OBJ)

# directive for the executable in the binary directory
$(LIBDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(LIBDIR)

.o.a :
    ar rv $@ $(OBJ)

.c.o :
    $(CC) -c $(CFLAGS) $<

.c.a :
    $(CC) -c $(CFLAGS) $<
    ar rv $@ $(OBJ)

# DO NOT DELETE THIS LINE - make depend uses it
eamate.o: eamate.c
eamate.o: /usr/ucblinclude/stdio.h
eamate.o: ../../..//include/params.h
eamate.o: ../../..//include/eamatestruct.h
general.o: general.c
general.o: /usr/ucblinclude/stdio.h
general.o: /usr/include/math.h
general.o: /usr/include/float.h
general.o: /usr/include/sys/ieeeefp.h
general.o: ../../..//include/params.h
general.o: ../../..//include/eamatestruct.h

```

```

/*
 * general.c
 * version 3.0
 * 09/08/93
 * by Natalie Willman
 *
 * This module defines general functions used by many of the .c
 * modules.
 *
 * Listing of Functions:
 * cr_browse_filename()
 * create_ein()
 * create_filename()
 * cr_detail_filename()
 * error_exit()
 * parse_ssn()
 * parse_name()
 * int_to_seq()
 * seq_to_int()
 * int_to_gram()
 * gram_to_int()
 * concat()
 * count_words()
 */

/* Include Files */
#include <stdio.h>
#include <math.h>
#include "params.h" /* general structure/parameter definitions & function prototypes */
#include "eamatestruct.h"

long binsearch_emplr();
long binsearch_mrn();

/*
 * error_exit()
 *
 * This function accepts as input an error message, and prints the
 * error message and exits
 *
 * Input: error message character string
 * Output: prints the error message, and exits
 */
void error_exit(message)
char message[];
{
    printf("%s\n", message);
    exit(1);
}

/*
 * parse_ssn
 *
 * This function takes a ssn field and parses it into
 * ngrams
 */
/*
 * Input: array containing the ssn, "ssn"
 * array to contain the parsed ssn, "parse_array"
 *
 * Output: array "parse_array" is filled with the ssn ngrams
 */

int parse_ssn(ssn, parse_array)
char ssn[];
char parse_array[ARRAY_SIZE][MAX_GRAM_SIZE+1];
{
    /* copy the first three digits to ngram #1
    strncpy(parse_array[0], ssn, 3);
    parse_array[0][3] = 0;

    /* copy the next two digits to ngram #2
    strncpy(parse_array[1], ssn+4, 2);
    parse_array[1][2] = 0;

    /* copy the last four digits to ngram #3
    strncpy(parse_array[2], ssn+7, 4);
    parse_array[2][4] = 0;

    return(3);
    */

/*
 * parse_name
 *
 * this function takes a name field and parses it into
 * ngrams of size GRAM_SIZE
 *
 * Input: array containing the name, "word"
 * array to contain the parsed name, "parse_array"
 *
 * Output: array "parse_array" is filled with the name ngrams
 */

int parse_name(word, parse_array)
char word[];
char parse_array[ARRAY_SIZE][MAX_GRAM_SIZE+1];
{
    int i = 0; /* counter variables
    int k;
    int j = 0;
    char valid_gram; /* bool value of gram validity (only alpha)
    char in_word = FALSE;

    /* While you are not at end of string ...
    while(word[i+GRAM_SIZE-1] != '\0')
    {
        /* pull out a gram from the string at location i, and check
        /* to see if all the members of gram are alpha chars
        if(!isalpha(word[i]))
        {
            strncpy(parse_array[i], word+i, GRAM_SIZE);
            valid_gram = TRUE;
            for(k = 0; k < GRAM_SIZE; k++)
                if(!isalpha(parse_array[i][k]))
                    parse_array[i][k] = toupper(parse_array[i][k]);
        }
    }
}

```

```

    else
        valid_gram = FALSE;
    /* If it is valid, then keep the gram, and increment loc */
    /* in word */
    if (valid_gram == TRUE)
    {
        in_word = TRUE;
        parse_array[j++][GRAM_SIZE] = 0;
    }
    else
    {
        if (in_word == TRUE)
            in_word = FALSE;
        else
        {
            for (k = 1; k < GRAM_SIZE; k++)
                parse_array[j][k] = ' ';
            parse_array[j++][GRAM_SIZE] = 0;
        }
    }

    if (OVERLAP == TRUE)
        i++;
    else
        i = 1 + GRAM_SIZE;
    else
        i++;
    }
    return(i);
}

/*
 * seq_to_int()
 *
 * this function will accept as input an ascii string which is a gram, and
 * will convert it to an integer value
 *
 * Input: string containing the gram, "gram"
 *
 * Output: integer representation of "gram" is returned
 */
seq_to_int(gram)
char gram[];
{
    int i, gramnum;
    int place;

    /* set the first place value to 26**SEQ_SIZE-1. This is the unit (like
    /* hundreds, tens, ones)
    place = (int) pow((double)26, (double)SEQ_SIZE-1);

    /* Initialize the gram num (int value to be returned)
    gramnum = 0;

    /* for each gram component ...
    for (i = 0; i < SEQ_SIZE; i++)
    {
        /* convert to an int value by adding 65 to convert to letter, and the
        /* multiply by unit value
        gramnum = gramnum + (gram[i] - 65) * place;

        /* reduce unit value
        place = place/26;
    }

    /* return integer value of gram
    return(gramnum);
}

/*
 * int_to_gram()
 *
 * this function will accept as input a integer value, and will convert it
 * to an ascii string to be used as a sequence number
 *
 * Input: integer value, "int_val"
 * string to hold seq, "gram"
 *
 * Output: "gram" contains the gram representation of "int_val"
 */
int_to_seq(int_val, gram)
int int_val;
char gram[];
{
    int j, remainder;
    int place, gramnum;

    /* Initialize the remainder to the entire value
    remainder = int_val;

    /* set the first place value to 26**SEQ_SIZE-1. This is the unit (like
    /* hundreds, tens, ones)
    place = (int) pow((double)26, (double) SEQ_SIZE-1);

    /* for each "place" value in the gram...
    for (j = 0; j < SEQ_SIZE-1; j++)
    {
        /* take the integer value of the remainder divided by the current place
        */

```



```

/* Input: Integer value, "Int_val"
 * string to hold gram, "gram"
 *
 * Output: "gram" contains the gram representation of "Int_val"
 */
int_to_gram(int_val, gram)
int int_val;
char gram[];
{
    int j, remainder;
    int place, gramnum;

    /* Initialize the remainder to the entire value
       remainder = int_val;

    /* set the first place value to 26**GRAM_SIZE-1. This is the unit (like
    /* hundreds, tens, ones)
    place = (int) pow((double)26, (double) GRAM_SIZE-1);

    /* for each "place" value in the gram...
    for(j = 0; j < GRAM_SIZE-1; j++)
    {
        /* take the integer value of the remainder divided by the current place
        gramnum = remainder/place;

        /* determine the remainder left from this division
        remainder = remainder - (gramnum * place);

        /* this gram is then equal to the integer div + 65 (to turn to a letter)
        gram[j] = gramnum+65;

        /* decrement the place value
        place = place/26;
    }

    /* set the last gram unit, and append an EOS
    gram[GRAM_SIZE-1] = remainder+65;
    gram[GRAM_SIZE] = 0;
}

/*
 * gram_to_int()
 *
 * this function will accept as input an ascii string which is a gram, and
 * will convert it to an integer value
 *
 * Input: string containing the gram, "gram"
 *
 * Output: Integer representation of "gram" is returned
 */
gram_to_int(gram)
char gram[];
{
    int i, gramnum;
    int place;

    /* set the first place value to 26**GRAM_SIZE-1. This is the unit (like
    /* hundreds, tens, ones)
    place = (int) pow((double)26, (double)GRAM_SIZE-1);

    /* Initialize the gram num (int value to be returned)
    gramnum = 0;

    /* for each gram component ...
    for(i = 0; i < GRAM_SIZE; i++)
    {
        /* convert to an int value by adding 65 to convert to letter, and the
        /* multiply by unit value
        gramnum = gramnum + (gram[i] - 65) * place;

        /* reduce unit value
        place = place/26;
    }

    /* return integer value of gram
    return(gramnum);
}

/*
 * cr_browse_filename()
 *
 * this function creates the filename for the browse data file -
 * YYYYBRW/BBB/EEEEEE.E to be compatible with DOS limitations
 * (E = EIN, Y = YEAR, B = Browse Location).
 *
 * Input: filename array to hold the name string, "filename"
 * array containing the year of the data, "year"
 * array containing the browse location, "browse_loc"
 * array containing the ein of the data, "ein"
 *
 * Output: array "filename" contains the browse filename
 */
void cr_browse_filename(filename, year, browse_loc, ein)
char filename[], year[], ein[];
char browse_loc[];
{
    if(strcmp(browse_loc, "0") == 0)
        sprintf(filename, "%sBRW/", year);
    else
        sprintf(filename, "%sBRW/%s/", year, browse_loc);
    strcat(filename, ein, 2);
    strcat(filename, ein+3, 6);
    strcat(filename, ".");
    strcat(filename, ein+9);
}

/*
 * create_ein()
 *
 * This function creates an ein from a browse filename
 *
 * Input: Array to contain ein, "ein"
 * Array containing the browse filename, "filename"
 * (NOTE: browse filename is without path)
 */
create_ein(ein, filename)
char ein[], filename[];
{

```

```

    strncpy(ein, filename, 2);
    ein[2] = 0;
    strcat(ein, "-");
    strncpyat(ein, filename+2, 6);
    strcat(ein, filename+9);
    ein[10] = 0;
}

/*
 *
 * * create_filename()
 *
 * * This function creates the base filename for the indexing files,
 * * (.idx and .dup)
 *
 * * Input: String to hold the base filename, filename
 * *         string containing the year, year
 * *         string containing the ein, ein
 *
 * * Output: filename contains the base filename
 *
 */
create_filename(filename, year, ein)
char filename[], year[], ein[];
{
    /* create base name for index files xxxx.yy-yyyyyyy where
    /* xxxx is the year and yy-yyyyyyy is the ein
    strncpy(filename, year);
    strcat(filename, "-");
    strcat(filename, ein);
}

/*
 *
 * * cr_detail_filename()
 *
 * * This function creates the filename for the detail files,
 * * stored on the optical disk
 *
 * * Input: String to hold the filename, filename
 * *         string containing the year, year
 * *         string containing the ein, ein
 * *         string containing the sequence string, seq
 * *         char containing the platter/side, platter_side
 *
 * * Output: filename contains the filename
 *
 */
cr_detail_filename(filename, year, ein, seq, platter_side)
char filename[], year[], ein[], seq[];
long platter_side;
{
    sprintf(filename, "%sDET/%d/", year, platter_side);
    strncpyat(filename, ein, 2);
    strncpyat(filename, ein+3, 6);
    strcat(filename, "-");
    strcat(filename, ein+9);
    strcat(filename, seq);
}

/*
 *
 * * function concat()
 *
 * * This function takes as input two character strings and returns the
 * * concatenation of these strings, while being non destructive to sl.
 * * The C function strcat() is destructive
 *
 * * Input: Two character strings, "s1" and "s2"
 * *         character string for concatenation, dest
 * * Output: dest string is filled with the concatenation of these two strings
 */
concat(dest, s1, s2)
char s1[], s2[], dest[];
{
    strcpy(dest, s1);
    strcat(dest, s2);
}

/*
 *
 * * count_words
 *
 * * This function counts the number of words available in
 * * a given string
 *
 * * note: a word is considered an alpha character string
 * *        that is two characters or longer.
 *
 * * Input: word string to count, "word"
 *
 * * Output: number of words in "word" string
 *
 */
count_words(word)
char word[];
{
    int i, wc;

    wc = 0;
    i = 0;
    do
    {
        /* move pointer until at beginning of a word */
        while(!isalpha(word[i])) && (word[i] != 0) )
            i++;

        /* If not at end of field, and if following char is also a
        /* letter, increment word count
        if(word[i] != 0)
            if( isalpha(word[i+1]) )
            {
                wc++;
                while(isalpha(word[i])) /* move ptr until at end of "word" */
                    i++;
            }
    } while(word[i] != 0); /* while not at end of field

    return(wc);
}

```

```

}

/*
 * search_all_emplr()
 *
 * This function will search through the 1991.employers.text file
 * and will find all employer header for a given year and ein, and
 * copy the header information into an output file that is passed
 * to the function
 */
search_all_emplr(year, ein, fileptr)
char year[], ein[];
FILE *fileptr;
{
    char namestring[FILENAME], namestring1[FILENAME];
    FILE *emlptr, *idxptr;
    long browse_offset;
    struct EAMATE W2EMPLR INFO employer_info;
    struct EMPLR_IDX emplr_idx;

    /* open the employer header master file, and the index to it */
    sprintf(namestring, "%s.employers.text", year);
    printf(namestring, "%s.employers.idx", year);
    emlptr = fopen(namestring, "r");
    idxptr = fopen(namestring, "r");
    if ( (emlptr == NULL) || (idxptr == NULL) )
    {
        printf("ERROR: Cannot open file %s or %s\n", namestring, namestring1);
        return(-1);
    }

    browse_offset = binsearch_emplr(ein, idxptr);
    if (browse_offset != -1)
    {
        /* If the ein exists, then read the record from the employer header
        /* file and print it to the output file. Continue until all headers
        /* for this ein have been copied
        fseek(idxptr, browse_offset, 0);
        fread(&emplr_idx, sizeof(struct EMPLR_IDX), 1, idxptr);
        while ( (strcmp(emplr_idx.ein, ein) == 0) && (!feof(idxptr)) )
        {
            seek_eamate_w2header_info(&employer_info, emplrptr,
            emplr_idx_offset);
            write_eamate_w2header_info(fileptr, &employer_info);
            fread(&emplr_idx, sizeof(struct EMPLR_IDX), 1, idxptr);
        }
        fclose(emplrptr);
        fclose(idxptr);
        return(1);
    }
    else
    {
        fclose(emplrptr);
        fclose(idxptr);
        return(-1);
    }
}

/*
 * binsearch_emplr()
 *
 * This function performs a binary search on the employer index file
 * for the ein entered (key)
 */
long binsearch_emplr(key, indexptr)
char key[];
FILE *indexptr;
{
    int left, right, index;
    struct EMPLR_IDX idx;
    long offset;

    /* Set up left and right bounds based upon the number of records */
    /* in the file
    left = 0;

    fseek(indexptr, 0, 2);
    right = (ftell(indexptr)/sizeof(struct EMPLR_IDX)) -1;

    /* While there are records between the left and right boundary.. */
    while (right > left)
    {
        /* Determine midpoint
        index = (left + right)/2;

        /* read the midpoint record
        fseek(indexptr, index*sizeof(struct EMPLR_IDX), 0);
        fread(&idx, sizeof(struct EMPLR_IDX), 1, indexptr);

        /* If the input key is greater than the midpoint, then change
        /* left and right bounds to only search upper half of data
        if (strcmp(key, idx.ein) > 0)
            left = index+1;

        /* Otherwise, change the bounds to only search the lower half
        /* of the data
        else
            right = index;
    }

    /* Once the left and right boundaries are the same, return the
    /* array index
    fseek(indexptr, left*sizeof(struct EMPLR_IDX), 0);
    offset = ftell(indexptr);
    fread(&idx, sizeof(struct EMPLR_IDX), 1, indexptr);
    if (strcmp(key, idx.ein) == 0)
        return(offset);
    else
        return(-1);
}

/*
 * search_seq_emplr()
 *
 * This function searches the employer master index file, and
 * looks for an employer header with a matching year, ein and
 * sequence number. The header is then printed to an output
 * file

```



```

/*
 */
search_seq_emplr(year, ein, seq, fileptr)
char year[], ein[], seq[];
FILE *fileptr;
{
    char namestring[FILENAME], namestring1[FILENAME];
    FILE *emlptr, *idxptr;
    long browse_offset;
    struct EAMATE_W2EMPLR_INFO employer_info;
    struct EMPLR_IDX emplr_idx;
    char done, found;

    /* open the employer header master file, and the index to it */
    sprintf(namestring, "%s.employers.text", year);
    sprintf(namestring1, "%s.employers.idx", year);
    emlptr = fopen(namestring, "r");
    idxptr = fopen(namestring1, "r");
    if ( (emlptr == NULL) || (idxptr == NULL) )
    {
        printf("ERROR: Cannot open file %s or %s\n", namestring, namestring1);
        return(-1);
    }

    browse_offset = binsearch_emplr(ein, idxptr);
    if (browse_offset != -1)
    {
        /* If the ein exists, then read the record from the employer header */
        /* file and print it to the output file. Continue until all headers */
        /* for this ein have been copied */
        fseek(idxptr, browse_offset, 0);
        fread(&emplr_idx, sizeof(struct EMPLR_IDX), 1, idxptr);
        done = FALSE;
        found = FALSE;
        while ( strcmp(emplr_idx.ein, ein) == 0 && (!done) )
        {
            seek_eamate_w2header_info(&employer_info, emlptr, emplr_idx.offset);
            if (strcmp(employer_info.seq_no, seq) == 0)
            {
                write_eamate_w2header_info(fileptr, &employer_info);
                done = TRUE;
                found = TRUE;
            }
            fread(&emplr_idx, sizeof(struct EMPLR_IDX), 1, idxptr);
            if (feof(idxptr))
                done = TRUE;
        }
        fclose(emlptr);
        fclose(idxptr);
        if (found == TRUE)
            return(1);
        else
        {
            printf("ERROR: Employer header not found for this sequence\n");
            return(-1);
        }
    }
    else
    {
        printf("ERROR: Employer header not found for this ein\n");
        return(-1);
    }
}

/*
 */
search_mrn(ein, year, mrn)
char ein[], year[];
char mrn[];
{
    FILE *idxptr; /* file ptrs */
    long loc;
    char namestring[FILENAME];

    /* open the file for the name b+ tree */
    cr_browse_filename(namestring, year, "0", ein);
    idxptr = fopen(namestring, "r");
    if (idxptr == NULL)
    {
        printf("ERROR: Cannot open file %s\n", namestring);
        return(-1);
    }
    else
    {
        /* Binary search for the mrn, and return an offset to the 1st record */
        /* containing it, a -1 if the mrn is not found */
        loc = binsearch_mrn(mrn, idxptr);
        fclose(idxptr);
        return(loc);
    }
}

/*
 */
binsearch_mrn()
{
    /* this function performs a binary search on the employee browse file
    * for the mrn entered (key)
    */
    long binsearch_mrn(key, idxptr)
    char key[];
    FILE *idxptr;
    {
        int left, right, index;
        struct EAMATE_W2EMPLR_BRW brw;
        long offset;

        /* Set up left and right bounds based upon the number of children */
        /* a node can have (1 + count)
        left = 0;

        fseek(idxptr, 0, 2);
        right = (ftell(idxptr)/sizeof(struct EAMATE_W2EMPLR_BRW)) -1;

        /* While there are records between the left and right boundary.. */
        while (right > left)

```

```

(
/* Determine midpoint
index = (left + right)/2;

fseek(indexptr, index*sizeof(struct EAMATE_W2EMPL_BRW), 0);
fread(sbrw, sizeof(struct EAMATE_W2EMPL_BRW), 1, indexptr);

/* If the input key is greater than the midpoint, then change
/* left and right bounds to only search upper half of data
if(strcasecmp(key, brw.mrn) > 0)
    left = index+1;
/* Otherwise, change the bounds to only search the lower half
/* of the data
else
    right = index;
)

/* Once the left and right boundaries are the same, return the
/* array index
fseek(indexptr, left*sizeof(struct EAMATE_W2EMPL_BRW), 0);
offset = ftell(indexptr);
fread(sbrw, sizeof(struct EAMATE_W2EMPL_BRW), 1, indexptr);
if(strcmp(key, brw.mrn) == 0)
    return(offset);
else
    return(-1);
}

/*
* translate_offsets
*
* This function takes a pointer to a list of match offsets, a browse data
* file and an output file, and a start location in the list of offsets.
* It will retrieve the browse information for the next SET_SIZE of browse
* offsets, and print the data to a file.
*
*/

translate_offsets(listptr, brwptr, reptr, start_loc)
FILE *listptr, *brwptr, *reptr;
long start_loc;
{
    int i, num;
    long offset[SET_SIZE];
    struct EAMATE_W2EMPL_BRW brw;

    fseek(listptr, start_loc * sizeof(long), 0);
    num = fread(offset, sizeof(long), SET_SIZE, listptr);
    if(num == 0)
    {
        printf("ERROR: No more matches available\n");
        return(-1);
    }
    else
    {
        for(i = 0; i < num; i++)
        {
            fseek(reptr, offset[i], 0);
            fread(sbrw, sizeof(struct EAMATE_W2EMPL_BRW), 1, reptr);
            fwrite(sbrw, sizeof(struct EAMATE_W2EMPL_BRW), 1, brwptr);
        }
        return(1);
    }
}

```

```

/*
 * eamate.c
 * version 3.0
 *
 * 09/08/93
 *
 * by Natalie Willman
 *
 * This module defines functions specific to the eamate data file.
 * Structure Definitions and function prototypes are in eamatestruct.h.
 * This module will interface with the B+tree module (btree.c),
 * the general functions module (general.c), the indexing module
 * (index.c) and the searching module (search.c). In addition, other
 * file modules can be linked in for use with other file types.
 *
 * List of Functions:
 *
 *   int read_eamate_w2header()
 *   int write_eamate_w2header()
 *   void display_eamate_w2header()
 *   void cp_eamate_w2header()
 *
 *   read_eamate_w2inter_total()
 *   write_eamate_w2inter_total()
 *   display_eamate_w2inter_tot()
 *
 *   read_eamate_w2final_total()
 *   write_eamate_w2final_total()
 *   display_eamate_w2final_total()
 *   void cp_eamate_w2final()
 *
 *   read_eamate_w2cumeln_total()
 *   write_eamate_w2cumeln_total()
 *   cp_eamate_w2cumeln()
 *   display_eamate_w2cumeln_tot()
 *
 *   int write_eamate_w2header_info()
 *   int seek_eamate_w2header_info()
 *   int read_eamate_w2header_info()
 *   void display_eamate_w2_header_info()
 *
 *   int read_eamate_w2employee_detail()
 *   int write_eamate_w2employee_detail()
 *   void display_eamate_w2employee_detail()
 *
 *   long write_eamate_w2employee_browse()
 *   int read_eamate_w2employee_browse()
 *   int seek_eamate_w2employee_browse()
 *   int extract_eamate_w2employee_browse()
 *   void display_eamate_w2employee_browse()
 */

/* Include Files */
#include <stdio.h>
#include "params.h" /* general structure/parameter definitions & function prototypes */
#include "eamatestruct.h" /* eamate structure definitions & function prototypes */

/*
 * read_eamate_w2header()
 *
 * This function reads an eamate w2 employer header record from a w2 data file
 */

```

```

/*
 * Input: Pointer to structure to hold data record, "pemplr"
 * Pointer to data file containing records, "data_file"
 *
 * Output: data is stored in "pemplr"
 */

int read_eamate_w2header(data_file, pemplr)
FILE *data_file;
struct EAMATE_W2EMPLR_HEADER *pemplr;
/* Read in the record
 * If (fread((char *)pemplr, sizeof(struct EAMATE_W2EMPLR_HEADER), 1, data_file) == 0)
 * return (FALSE);
 * else
 * return (TRUE);
 */

/*
 * write_eamate_w2header()
 *
 * This function writes an eamate w2 employer header record to a file
 *
 * Input: Pointer to structure containing the data record, "pemplr"
 * Pointer to data file to write record, "data_file"
 *
 * Output: data is written to "data_file"
 */

int write_eamate_w2header(data_file, pemplr)
FILE *data_file;
struct EAMATE_W2EMPLR_HEADER *pemplr;
{
    char typecd = MATE_W2EH;

    fwrite((char *)&typecd, sizeof(char), 1, data_file);
    if (fwrite((char *)pemplr, sizeof(struct EAMATE_W2EMPLR_HEADER), 1, data_file) == 0)
        return (FALSE);
    else
        return (TRUE);
}

/*
 * display_eamate_w2header()
 *
 * This function displays an employer header record
 *
 * Input: Pointer to structure containing employer header record, "pemplr"
 *
 * Output: struct "pemplr" is displayed
 */

void display_eamate_w2header(pemplr)
struct EAMATE_W2EMPLR_HEADER *pemplr;
{
    printf("EAMATE W2 Employer Header\n");
    printf("ein = %s\n", pemplr->ein);
    printf("est = %s\n", pemplr->est);
    printf("rep year = %s\n", pemplr->rpt_yr);
}

```



```

printf("proc year = %s\n", pemplr->proc_yr);
printf("tape lib = %s\n", pemplr->tape_lib_num);
printf("type_emplr = %s\n", pemplr->type_emplr);
printf("name code = %s\n", pemplr->name_code);
printf("other_ein = %s\n", pemplr->other_ein);
printf("mrn = %s\n", pemplr->mrn);
printf("end mrn = %s\n", pemplr->end_mrn);
printf("seqno = %s\n", pemplr->seq_no);

printf("name = %s\n", pemplr->name);
printf("addr = %s\n", pemplr->street_add);
printf("city = %s\n", pemplr->city);
printf("st = %s\n", pemplr->state);
printf("zip = %s\n", pemplr->zip_code);
printf("platter/side = %d\n", pemplr->platter_side);
printf("num recs = %d\n", pemplr->num_recs);
printf("final_offset = %d\n", pemplr->final_offset);
printf("cum_offset = %d\n", pemplr->cum_offset);
getchar();
}

/*
 * output_eamate_w2header()
 *
 * This function outputs an employer header record to a file
 *
 * Input: Pointer to structure containing employer header record, "pemplr"
 *
 * file pointer, "file"
 *
 * Output: struct "pemplr" is displayed
 */
output_eamate_w2header(pemplr, file)
struct EAMATE_W2EMPLR_HEADER *pemplr;
FILE *file;
{
    fprintf(file, "EAMATE W2 Employer Header\n");
    fprintf(file, "ein = %s\n", pemplr->ein);
    fprintf(file, "est = %s\n", pemplr->est);
    fprintf(file, "rep year = %s\n", pemplr->rpt_yr);
    fprintf(file, "proc year = %s\n", pemplr->proc_yr);
    fprintf(file, "tape lib = %s\n", pemplr->tape_lib_num);
    fprintf(file, "type_emplr = %s\n", pemplr->type_emplr);
    fprintf(file, "name code = %s\n", pemplr->name_code);
    fprintf(file, "other_ein = %s\n", pemplr->other_ein);
    fprintf(file, "mrn = %s\n", pemplr->mrn);
    fprintf(file, "end mrn = %s\n", pemplr->end_mrn);
    fprintf(file, "seqno = %s\n", pemplr->seq_no);

    fprintf(file, "name = %s\n", pemplr->name);
    fprintf(file, "addr = %s\n", pemplr->street_add);
    fprintf(file, "city = %s\n", pemplr->city);
    fprintf(file, "st = %s\n", pemplr->state);
    fprintf(file, "zip = %s\n", pemplr->zip_code);
    fprintf(file, "platter/side = %d\n", pemplr->platter_side);
    fprintf(file, "num recs = %d\n", pemplr->num_recs);
    fprintf(file, "final_offset = %d\n", pemplr->final_offset);
    fprintf(file, "cum_offset = %d\n", pemplr->cum_offset);
}

/*
 * cp_eamate_w2header()
 *
 * This function copies the data contained in a EAMATE_W2EMPLR_HEADER
 * structure into a EAMATE_W2EMPLR_INFO structure
 *
 * Input: Pointer to the employer header structure, "pemplr"
 *
 * Pointer to employer header info structure, "pemplr_info"
 *
 * Output: data in header structure is copied to header info structure
 */
void cp_eamate_w2header(pemplr, pemplr_info)
struct EAMATE_W2EMPLR_HEADER *pemplr;
struct EAMATE_W2EMPLR_INFO *pemplr_info;
{
    strcpy(pemplr_info->ein, pemplr->ein);
    strcpy(pemplr_info->est, pemplr->est);
    strcpy(pemplr_info->rpt_yr, pemplr->rpt_yr);
    strcpy(pemplr_info->proc_yr, pemplr->proc_yr);
    strcpy(pemplr_info->tape_lib_num, pemplr->tape_lib_num);
    strcpy(pemplr_info->type_emplr, pemplr->type_emplr);
    strcpy(pemplr_info->name_code, pemplr->name_code);
    strcpy(pemplr_info->other_ein, pemplr->other_ein);
    strcpy(pemplr_info->mrn, pemplr->mrn);
    strcpy(pemplr_info->end_mrn, pemplr->end_mrn);
    strcpy(pemplr_info->seq_no, pemplr->seq_no);

    strcpy(pemplr_info->name, pemplr->name);
    strcpy(pemplr_info->street_add, pemplr->street_add);
    strcpy(pemplr_info->city, pemplr->city);
    strcpy(pemplr_info->state, pemplr->state);
    strcpy(pemplr_info->zip_code, pemplr->zip_code);
    pemplr_info->platter_side = pemplr->platter_side;
    pemplr_info->num_recs = pemplr->num_recs;
    pemplr_info->cflag = 0;
}

/*
 * read_eamate_w2inter_total()
 *
 * This function reads an eamate W2 intermediate total record from a W2 data file
 *
 * Input: Pointer to structure to hold data record, "pinter_total"
 *
 * Pointer to data file containing records, "data_file"
 *
 * Output: data is stored in "pinter_total"
 */
int read_eamate_w2inter_total(data_file, pinter_total)
FILE *data_file;
struct EAMATE_W2INTERMED_TOT *pinter_total;
{
    /* Read in the record
    if (fread((char *)pinter_total, sizeof(struct EAMATE_W2INTERMED_TOT), 1, data_file) ==
    0)
        return (FALSE);
    else
        return (TRUE);
    */
}

```

```

/*
 * write_eamate_w2inter_total()
 *
 * This function writes an eamate W2 intermediate total record to a file
 *
 * Input: Pointer to structure containing the data record, "pinter_total"
 *         Pointer to data file to write record, "data_file"
 *
 * Output: data is written to "data_file"
 */

Int write_eamate_w2inter_total(data_file, pinter_total)
FILE *data_file;
struct EAMATE_W2INTERMED_TOT *pinter_total;
{
    char typeid = MATE_W2IT;

    fwrite((char *)typeid, sizeof(char), 1, data_file);
    if(fwrite((char *)pinter_total, sizeof(struct EAMATE_W2INTERMED_TOT), 1, data_file) == 0)
        return(FALSE);
    else
        return(TRUE);
}

/*
 * display_eamate_w2inter_tot()
 *
 * This record prints an intermediate total record on the
 * screen
 *
 * Input: structure containing the intermediate total record
 *
 * Output: The intermediate total record is printed
 */

display_eamate_w2inter_tot(empl_rec)
struct EAMATE_W2INTERMED_TOT empl_rec;
{
    printf("Intermediate Total\n");
    printf("proc wages = %s\n", empl_rec.proc_wages);
    printf("rep wages = %s\n", empl_rec.rep_wages);
    printf("proc tips = %s\n", empl_rec.proc_tips);
    printf("rep tips = %s\n", empl_rec.rep_tips);
    printf("proc other = %s\n", empl_rec.proc_other);
    printf("rep other = %s\n", empl_rec.rep_other);
    printf("proc fed tax = %s\n", empl_rec.proc_fed_tax);
    printf("rep fed tax = %s\n", empl_rec.rep_fed_tax);
    printf("proc fica tax = %s\n", empl_rec.proc_fica_tax);
    printf("rep fica tax = %s\n", empl_rec.rep_fica_tax);
    printf("proc adv_earn_inc = %s\n", empl_rec.proc_earn_inc);
    printf("rep adv_earn_inc = %s\n", empl_rec.rep_earn_inc);
    printf("proc defcomp = %s\n", empl_rec.proc_defcomp);
    printf("rep defcomp = %s\n", empl_rec.rep_defcomp);
    printf("proc nonqual = %s\n", empl_rec.proc_nonqual);
    printf("rep nonqual = %s\n", empl_rec.rep_nonqual);
    printf("ctrl_no = %s\n", empl_rec.ctrl_no);

    printf("proc med_wages = %s\n", empl_rec.proc_med_wages);
    printf("rep med_wages = %s\n", empl_rec.rep_med_wages);
    printf("proc med_tax = %s\n", empl_rec.proc_med_tax);
    printf("rep med_tax = %s\n", empl_rec.rep_med_tax);
}

/*
 * read_eamate_w2final_total()
 *
 * This function reads an eamate W2 final total record from a W2 data file
 *
 * Input: Pointer to structure to hold data record, "pfinal_total"
 *         Pointer to data file containing records, "data_file"
 *
 * Output: data is stored in "pfinal_total"
 */

int read_eamate_w2final_total(data_file, pfinal_total)
FILE *data_file;
struct EAMATE_W2FINAL_TOT *pfinal_total;
{
    /* Read in the record

```

```

    if(fread((char *)pfinal_total, sizeof(struct EAMATE_W2FINAL_TOT), 1, data_file) == 0)
        return(FALSE);
    else
        return(TRUE);
}

/*
 * write_eamate_w2final_total()
 *
 * This function writes an eamate w2 final total record to a file
 *
 * Input: Pointer to structure containing the data record, "pfinal_total"
 *        Pointer to data file to write record, "data_file"
 *
 * Output: data is written to "data_file"
 */
int write_eamate_w2final_total(data_file, pfinal_total)
FILE *data_file;
struct EAMATE_W2FINAL_TOT *pfinal_total;
{
    char typeid = MATE_W2FT;

    fwrite((char *)typeid, sizeof(char), 1, data_file);
    if(fwrite((char *)pfinal_total, sizeof(struct EAMATE_W2FINAL_TOT), 1, data_file) == 0)
        return(FALSE);
    else
        return(TRUE);
}

/*
 * display_eamate_w2final_tot()
 *
 * This record prints a final total record on the
 * screen
 *
 * Input: structure containing the final total record
 *
 * Output: The final total record is printed
 */
display_eamate_w2final_tot(empl_rec)
struct EAMATE_W2FINAL_TOT empl_rec;
{
    printf("Final Total\n");
    printf("proc wages = %s\n", empl_rec.proc_wages);
    printf("rep wages = %s\n", empl_rec.rep_wages);
    printf("proc tips = %s\n", empl_rec.proc_tips);
    printf("rep tips = %s\n", empl_rec.rep_tips);
    printf("proc other = %s\n", empl_rec.proc_other);
    printf("rep other = %s\n", empl_rec.rep_other);
    printf("proc fed tax = %s\n", empl_rec.proc_fed_tax);
    printf("rep fed tax = %s\n", empl_rec.rep_fed_tax);
    printf("proc fica tax = %s\n", empl_rec.proc_fica_tax);
    printf("rep fica tax = %s\n", empl_rec.rep_fica_tax);
    printf("proc adv_earn_inc = %s\n", empl_rec.proc_earn_inc);
    printf("rep adv_earn_inc = %s\n", empl_rec.rep_earn_inc);
    printf("proc items = %s\n", empl_rec.proc_items);
    printf("rep items = %s\n", empl_rec.rep_items);

    printf("proc defcomp = %s\n", empl_rec.proc_defcomp);
    printf("rep defcomp = %s\n", empl_rec.rep_defcomp);
    printf("proc nonqual = %s\n", empl_rec.proc_nonqual);
    printf("rep nonqual = %s\n", empl_rec.rep_nonqual);
    printf("proc med_wages = %s\n", empl_rec.proc_med_wages);
    printf("rep med_wages = %s\n", empl_rec.rep_med_wages);
    printf("proc med_tax = %s\n", empl_rec.proc_med_tax);
    printf("rep med_tax = %s\n", empl_rec.rep_med_tax);
}

/*
 * cp_eamate_w2final()
 *
 * This function copies the data contained in a EAMATE_W2FINAL_TOT
 * structure into a EAMATE_W2EMPLR_INFO structure
 */
printf("rep items = %s\n", empl_rec.rep_items);
printf("proc defcomp = %s\n", empl_rec.proc_defcomp);
printf("rep defcomp = %s\n", empl_rec.rep_defcomp);
printf("proc nonqual = %s\n", empl_rec.proc_nonqual);
printf("rep nonqual = %s\n", empl_rec.rep_nonqual);
printf("proc med_wages = %s\n", empl_rec.proc_med_wages);
printf("rep med_wages = %s\n", empl_rec.rep_med_wages);
printf("proc med_tax = %s\n", empl_rec.proc_med_tax);
printf("rep med_tax = %s\n", empl_rec.rep_med_tax);
}

/*
 * cp_eamate_w2final()
 *
 * This function copies the data contained in a EAMATE_W2FINAL_TOT
 * structure into a EAMATE_W2EMPLR_INFO structure
 */

```



```

* Input:  data structure containing final total, "final_rec"
*         data structure containing employer info, "pemplr_info"
*
* Output: final total data is stored in the employer info structure
*/

void cp_eamate_w2final(final_rec, pemplr_info)
struct_EAMATE_W2FINAL_TOT final_rec;
struct_EAMATE_W2EMPLR_INFO *pemplr_info;
{
    strcpy(pemplr_info->proc_wages, final_rec.proc_wages);
    strcpy(pemplr_info->rep_wages, final_rec.rep_wages);
    strcpy(pemplr_info->proc_tips, final_rec.proc_tips);
    strcpy(pemplr_info->rep_tips, final_rec.rep_tips);
    strcpy(pemplr_info->proc_other, final_rec.proc_other);
    strcpy(pemplr_info->rep_other, final_rec.rep_other);
    strcpy(pemplr_info->proc_fed_tax, final_rec.proc_fed_tax);
    strcpy(pemplr_info->rep_fed_tax, final_rec.rep_fed_tax);
    strcpy(pemplr_info->proc_fica_tax, final_rec.proc_fica_tax);
    strcpy(pemplr_info->rep_fica_tax, final_rec.rep_fica_tax);
    strcpy(pemplr_info->proc_earn_inc, final_rec.proc_earn_inc);
    strcpy(pemplr_info->rep_earn_inc, final_rec.rep_earn_inc);
    strcpy(pemplr_info->proc_items, final_rec.proc_items);
    strcpy(pemplr_info->rep_items, final_rec.rep_items);
    strcpy(pemplr_info->proc_defcomp, final_rec.proc_defcomp);
    strcpy(pemplr_info->rep_defcomp, final_rec.rep_defcomp);
    strcpy(pemplr_info->proc_nonqual, final_rec.proc_nonqual);
    strcpy(pemplr_info->rep_nonqual, final_rec.rep_nonqual);
    strcpy(pemplr_info->proc_med_wages, final_rec.proc_med_wages);
    strcpy(pemplr_info->rep_med_wages, final_rec.rep_med_wages);
    strcpy(pemplr_info->proc_med_tax, final_rec.proc_med_tax);
    strcpy(pemplr_info->rep_med_tax, final_rec.rep_med_tax);
}

/*
* read_eamate_w2cumeln_total()
*
* This function reads an eamate W2 cumulative ein total record from a W2 data file
*
* Input:  Pointer to structure to hold data record, "pcumein_total"
*         Pointer to data file containing records, "data_file"
*
* Output: data is stored in "pcumein_total"
*/

int read_eamate_w2cumeln_total(data_file, pcumein_total)
FILE *data_file;
struct_EAMATE_W2CUMEIN_TOT *pcumein_total;
{
    /* Read in the record
    if(fread((char *)pcumein_total, sizeof(struct_EAMATE_W2CUMEIN_TOT), 1, data_file) ==
    0)
        return(FALSE);
    else
        return(TRUE);
    */

    /* write_eamate_w2cumeln_total()
    */
}

```

```

* This function writes an eamate W2 cum ein total record to a file
*
* Input:  Pointer to structure containing the data record, "pcumein_total"
*         Pointer to data file to write record, "data_file"
*
* Output: data is written to "data_file"
*/

int write_eamate_w2cumeln_total(data_file, pcumein_total)
FILE *data_file;
struct_EAMATE_W2CUMEIN_TOT *pcumein_total;
{
    char typeid = MATE_W2CE;

    fwrite((char *)typeid, sizeof(char), 1, data_file);
    if(fwrite((char *)pcumein_total, sizeof(struct_EAMATE_W2CUMEIN_TOT), 1, data_file) == 0)
        return(FALSE);
    else
        return(TRUE);
}

/*
* display_eamate_w2cumeln_tot()
*
* This record prints a cum ein record on the
* screen
*
* Input:  structure containing the final total record
*
* Output: The final total record is printed
*/

display_eamate_w2cumeln_tot(empl_rec)
struct_EAMATE_W2CUMEIN_TOT empl_rec;
{
    printf("CUM EIN\n");
    printf("proc wages = %s\n", empl_rec.proc_wages);
    printf("proc tips = %s\n", empl_rec.proc_tips);
    printf("proc other = %s\n", empl_rec.proc_other);
    printf("proc fed_tax = %s\n", empl_rec.proc_fed_tax);
    printf("proc fica_tax = %s\n", empl_rec.proc_fica_tax);
    printf("proc adv_earn_inc = %s\n", empl_rec.proc_earn_inc);
    printf("proc items = %s\n", empl_rec.proc_items);
    getchar();
}

/*
* output_eamate_w2cumeln_tot()
*
* This record prints a cum ein record on the
* screen
*
* Input:  structure containing the final total record
*
* Output: The final total record is printed
*/

output_eamate_w2cumeln_tot(empl_rec, file)
struct_EAMATE_W2CUMEIN_TOT empl_rec;

```

```

FILE *file;
{
    fprintf(file, "CUM EIN\n");
    fprintf(file, "proc wages = %s\n", empl_rec.proc.wages);
    fprintf(file, "proc tips = %s\n", empl_rec.proc.tips);
    fprintf(file, "proc other = %s\n", empl_rec.proc.other);
    fprintf(file, "proc fed tax = %s\n", empl_rec.proc.fed_tax);
    fprintf(file, "proc fica tax = %s\n", empl_rec.proc.fica_tax);
    fprintf(file, "proc adv_earn_inc = %s\n", empl_rec.proc.earn_inc);
    fprintf(file, "proc items = %s\n", empl_rec.proc.items);
}

/*
 * cp_eamate_w2cum()
 *
 * This function copies the data contained in a EAMATE_W2CUMEIN_TOT
 * structure into a EAMATE_W2EMPLR_INFO structure
 *
 * Input:  data structure containing cum ein info, "cumeln"
 *         data structure containing employer info, "pemplr_info"
 *
 * Output: Cum ein info is stored in employer info structure
 */
void cp_eamate_w2cum(cum_ein, pemplr_info)
struct EAMATE_W2CUMEIN_TOT cum_ein;
struct EAMATE_W2EMPLR_INFO *pemplr_info;
{
    pemplr_info->cflag = 1;
    strcpy(pemplr_info->cproc.wages, cum_ein.proc.wages);
    strcpy(pemplr_info->cproc.tips, cum_ein.proc.tips);
    strcpy(pemplr_info->cproc.other, cum_ein.proc.other);
    strcpy(pemplr_info->cproc.fed_tax, cum_ein.proc.fed_tax);
    strcpy(pemplr_info->cproc.fica_tax, cum_ein.proc.fica_tax);
    strcpy(pemplr_info->cproc.earn_inc, cum_ein.proc.earn_inc);
    strcpy(pemplr_info->cproc.items, cum_ein.proc.items);
}

/*
 * write_eamate_w2header_info()
 *
 * This function writes an employer info record to a file
 *
 * Input:  Pointer to structure containing employer info record, "pemplr_info"
 *         Pointer to data file to write records, "data_file"
 *
 * Output: file "data_file" is updated
 */
int write_eamate_w2header_info(data_file, pemplr_info)
FILE *data_file;
struct EAMATE_W2EMPLR_INFO *pemplr_info;
{
    if (fwrite((char *)pemplr_info, sizeof(struct EAMATE_W2EMPLR_INFO), 1, data_file) == 0)
        return(FALSE);
    else
        return(TRUE);
}

/*
 * display_eamate_w2header_info()
 *
 * This function displays an employer header information record
 *
 * Input:  Pointer to structure containing data record, "pemplr_info"
 */

```

```

*
* Output: struct "pemplr_info" is displayed
*
*/

void display_eamate_w2header_info(pemplr_info)
{
    struct EAMATE_W2EMPLR_INFO *pemplr_info;

    printf("ein = %s\n", pemplr_info->ein);
    printf("est = %s\n", pemplr_info->est);
    printf("rep year = %s\n", pemplr_info->rpt_yr);
    printf("proc year = %s\n", pemplr_info->proc_yr);
    printf("tape lib = %s\n", pemplr_info->tape_lib_num);
    printf("type emplr = %s\n", pemplr_info->type_emplr);
    printf("name code = %s\n", pemplr_info->name_code);
    printf("other_ein = %s\n", pemplr_info->other_ein);
    printf("mrn = %s\n", pemplr_info->mrn);
    printf("end mrn = %s\n", pemplr_info->end_mrn);
    printf("seq no = %s\n", pemplr_info->seq_no);

    printf("name = %s\n", pemplr_info->name);
    printf("addr = %s\n", pemplr_info->street_add);
    printf("city = %s\n", pemplr_info->city);
    printf("st = %s\n", pemplr_info->state);
    printf("zip = %s\n", pemplr_info->zip_code);
    printf("platter/side = %d\n", pemplr_info->platter_side);
    printf("num recs = %d\n", pemplr_info->num_rec);
    printf("initials = %d\n", pemplr_info->initials);
    printf("browse start = %d\n", pemplr_info->browse_start);

    printf("proc wages = %s\n", pemplr_info->proc_wages);
    printf("rep wages = %s\n", pemplr_info->rep_wages);
    printf("proc tips = %s\n", pemplr_info->proc_tips);
    printf("rep tips = %s\n", pemplr_info->rep_tips);
    printf("proc other = %s\n", pemplr_info->proc_other);
    printf("rep other = %s\n", pemplr_info->rep_other);
    printf("proc fed tax = %s\n", pemplr_info->proc_fed_tax);
    printf("rep fed tax = %s\n", pemplr_info->rep_fed_tax);
    printf("proc fica tax = %s\n", pemplr_info->proc_fica_tax);
    printf("rep fica tax = %s\n", pemplr_info->rep_fica_tax);
    printf("proc earn inc = %s\n", pemplr_info->proc_earn_inc);
    printf("rep earn inc = %s\n", pemplr_info->rep_earn_inc);
    printf("proc items = %s\n", pemplr_info->proc_items);
    printf("rep items = %s\n", pemplr_info->rep_items);

    printf("proc defcomp = %s\n", pemplr_info->proc_defcomp);
    printf("rep defcomp = %s\n", pemplr_info->rep_defcomp);
    printf("proc nonqual = %s\n", pemplr_info->proc_nonqual);
    printf("rep nonqual = %s\n", pemplr_info->rep_nonqual);
    printf("proc med wages = %s\n", pemplr_info->proc_med_wages);
    printf("rep med wages = %s\n", pemplr_info->rep_med_wages);
    printf("proc med tax = %s\n", pemplr_info->proc_med_tax);
    printf("rep med tax = %s\n", pemplr_info->rep_med_tax);

    if (pemplr_info->cflag == 1)
    {
        printf("cproc wages = %s\n", pemplr_info->cproc_wages);
        printf("cproc tips = %s\n", pemplr_info->cproc_tips);
        printf("cproc other = %s\n", pemplr_info->cproc_other);
        printf("cproc fed tax = %s\n", pemplr_info->cproc_fed_tax);
        printf("cproc fica tax = %s\n", pemplr_info->cproc_fica_tax);
        printf("cproc earn inc = %s\n", pemplr_info->cproc_earn_inc);
        printf("cproc items = %s\n", pemplr_info->cproc_items);
    }
}

/*
* output_eamate_w2header_info()
*
* This function displays an employer header information record
*
* Input: Pointer to structure containing data record, "pemplr_info"
*
* Output: struct "pemplr_info" is displayed
*/
*/

out_put_eamate_w2header_info(pemplr_info, file)
    struct EAMATE_W2EMPLR_INFO *pemplr_info;
    FILE *file;
{
    fprintf(file, "ein = %s\n", pemplr_info->ein);
    fprintf(file, "est = %s\n", pemplr_info->est);
    fprintf(file, "rep year = %s\n", pemplr_info->rpt_yr);
    fprintf(file, "proc year = %s\n", pemplr_info->proc_yr);
    fprintf(file, "tape lib = %s\n", pemplr_info->tape_lib_num);
    fprintf(file, "type emplr = %s\n", pemplr_info->type_emplr);
    fprintf(file, "name code = %s\n", pemplr_info->name_code);
    fprintf(file, "other_ein = %s\n", pemplr_info->other_ein);
    fprintf(file, "mrn = %s\n", pemplr_info->mrn);
    fprintf(file, "end mrn = %s\n", pemplr_info->end_mrn);
    fprintf(file, "seq no = %s\n", pemplr_info->seq_no);

    fprintf(file, "name = %s\n", pemplr_info->name);
    fprintf(file, "addr = %s\n", pemplr_info->street_add);
    fprintf(file, "city = %s\n", pemplr_info->city);
    fprintf(file, "st = %s\n", pemplr_info->state);
    fprintf(file, "zip = %s\n", pemplr_info->zip_code);
    fprintf(file, "platter/side = %d\n", pemplr_info->platter_side);
    fprintf(file, "num recs = %d\n", pemplr_info->num_rec);
    fprintf(file, "initials = %d\n", pemplr_info->initials);
    fprintf(file, "browse start = %d\n", pemplr_info->browse_start);

    fprintf(file, "proc wages = %s\n", pemplr_info->proc_wages);
    fprintf(file, "rep wages = %s\n", pemplr_info->rep_wages);
    fprintf(file, "proc tips = %s\n", pemplr_info->proc_tips);
    fprintf(file, "rep tips = %s\n", pemplr_info->rep_tips);
    fprintf(file, "proc other = %s\n", pemplr_info->proc_other);
    fprintf(file, "rep other = %s\n", pemplr_info->rep_other);
    fprintf(file, "proc fed tax = %s\n", pemplr_info->proc_fed_tax);
    fprintf(file, "rep fed tax = %s\n", pemplr_info->rep_fed_tax);
    fprintf(file, "proc fica tax = %s\n", pemplr_info->proc_fica_tax);
    fprintf(file, "rep fica tax = %s\n", pemplr_info->rep_fica_tax);
    fprintf(file, "proc earn inc = %s\n", pemplr_info->proc_earn_inc);
    fprintf(file, "rep earn inc = %s\n", pemplr_info->rep_earn_inc);
    fprintf(file, "proc items = %s\n", pemplr_info->proc_items);
    fprintf(file, "rep items = %s\n", pemplr_info->rep_items);

    fprintf(file, "proc defcomp = %s\n", pemplr_info->proc_defcomp);
    fprintf(file, "rep defcomp = %s\n", pemplr_info->rep_defcomp);
    fprintf(file, "proc nonqual = %s\n", pemplr_info->proc_nonqual);
    fprintf(file, "rep nonqual = %s\n", pemplr_info->rep_nonqual);
    fprintf(file, "proc med wages = %s\n", pemplr_info->proc_med_wages);
    fprintf(file, "rep med wages = %s\n", pemplr_info->rep_med_wages);
    fprintf(file, "proc med tax = %s\n", pemplr_info->proc_med_tax);
    fprintf(file, "rep med tax = %s\n", pemplr_info->rep_med_tax);
}

```



```

/* write the record
if(fwrite((char *)rec, sizeof(struct EAMATE_W2EMPL_DETAIL), 1, data_file) == 0)
    return(FALSE);
else
    return(offset);
}

/*
* display_eamate_w2employee_detail()
*
* This record prints an employee detail record on the
* screen
*
* Input: structure containing the employee record information, "empl_rec"
*
* Output: The employee record is printed
*/

void display_eamate_w2employee_detail(empl_rec)
struct EAMATE_W2EMPL_DETAIL empl_rec;
{
    printf("Employee Record\n");
    printf("mrn = %s\n", empl_rec.mrn);
    printf("ssn = %s\n", empl_rec.ssn);
    printf("name = %s\n", empl_rec.name);
    printf("pens = %s\n", empl_rec.pens);
    printf("def = %s\n", empl_rec.defcomp);
    printf("wages = %s\n", empl_rec.wages);
    printf("tips = %s\n", empl_rec.tips);
    printf("other = %s\n", empl_rec.other);
    printf("fed tax = %s\n", empl_rec.fedtax);
    printf("fica tax = %s\n", empl_rec.fica_tax);
    printf("adv_earn_inc = %s\n", empl_rec.adv_earn_inc);
    printf("med_wages = %s\n", empl_rec.med_wages);
    printf("med_tax = %s\n", empl_rec.med_tax);
    printf("ctrl_no = %s\n", empl_rec.ctrl_no);

    printf("street add = %s\n", empl_rec.street_add);
    printf("dep_care = %s\n", empl_rec.dep_care);
    printf("alloc_tips = %s\n", empl_rec.alloc_tips);
    printf("grp_insur = %s\n", empl_rec.grp_insur);
    printf("uncoil_fical_tax = %s\n", empl_rec.uncoil_fical_tax);

    printf("city = %s\n", empl_rec.city);
    printf("state = %s\n", empl_rec.state);
    printf("zip_code = %s\n", empl_rec.zip_code);
    printf("defcomp = %s\n", empl_rec.defcomp);
    printf("sta = %s\n", empl_rec.sta);
    printf("fr_ben = %s\n", empl_rec.fr_ben);
    printf("ndsec = %s\n", empl_rec.ndsec);
    printf("nqnot = %s\n", empl_rec.nqnot);
    getchar();
}

/*
* output_eamate_w2employee_detail()
*
* This record prints an employee detail record on the
* screen
*
* Input: structure containing the employee record information, "empl_rec"
*/

```

```

    loc = ftell(data_fll);
    if(fwrite((char *)rec, sizeof(struct EAMATE_W2EMPL_BRW), 1, data_fll) == 0)
        return(FALSE);
    else
        return(loc);
}

/*
 * extract_eamate_w2employee_browse()
 * This function stores the browse fields of a EAMATE_W2EMPL_DETAIL
 * structure into a structure of type EAMATE_W2EMPL_BRW.
 *
 * Input: Pointer to structure to hold data record, "prec"
 * Pointer to structure containing detail record, "empl_rec"
 * Char string containing the sequence number of the report
 * pointer to the offset of the full record in the detail file
 * wage type (original, corrected, etc.)
 *
 * Output: data is stored in "prec"
 */
int extract_eamate_w2employee_browse(empl_rec, prec, seq, detail_offset, wage_type)
struct EAMATE_W2EMPL_BRW *prec;
struct EAMATE_W2EMPL_DETAIL empl_rec;
char seq[], wage_type[];
long detail_offset;
{
    /* Copy the browse portion of the record to the browse structure */
    strcpy(prec->ssn, empl_rec.ssn);
    strcpy(prec->name, empl_rec.name);
    strcpy(prec->wages, empl_rec.wages);
    strcpy(prec->tips, empl_rec.tips);
    strcpy(prec->fica_tax, empl_rec.fica_tax);
    strcpy(prec->other, empl_rec.other);
    strcpy(prec->mrn, empl_rec.mrn);
    strcpy(prec->seq_no, seq);
    strcpy(prec->wage_type, wage_type);
    prec->record_loc = detail_offset;
}

/*
 * read_eamate_w2employee_browse()
 * This function reads a record of type EAMATE_W2EMPL_BRW from
 * a data file.
 *
 * Input: Pointer to structure holding data record, "prec"
 * File Pointer to the data file, "data_fll"
 *
 * Output: Data structure is filled
 */
int read_eamate_w2employee_browse(prec, data_fll)
struct EAMATE_W2EMPL_BRW *prec;
FILE *data_fll;
{
    /* read the record
    if(fread((char *)prec, sizeof(struct EAMATE_W2EMPL_BRW), 1, data_fll) == 0)
    */
}

/*
 * write_eamate_w2employee_browse()
 * This function writes a record of type EAMATE_W2EMPL_BRW to
 * a data file.
 *
 * Input: Pointer to structure holding data record, "rec"
 * File Pointer to the data file, "data_fll"
 *
 * Output: Data file is updated
 */
long write_eamate_w2employee_browse(rec, data_fll)
struct EAMATE_W2EMPL_BRW *rec;
FILE *data_fll;
{
    long loc;

    /* Record location of the record in the browse file, write the */
    /* record and return the browse file location (this goes in */
    /* the index */
}

```

```

    return(FALSE);
else
    return(TRUE);
}

/*
 * seek_eamate_w2employee_browse()
 *
 * This function seeks to and reads a record of type EAMATE_W2EMPL_BRW
 * from a data file.
 *
 * Input: Pointer to structure holding data record, "rec"
 * File Pointer to the data file, "data_fil"
 * Location from which to read the browse data, "loc"
 *
 * Output: Data structure is filled
 */
int seek_eamate_w2employee_browse(prec, data_fil, loc)
struct EAMATE_W2EMPL_BRW *prec;
FILE *data_fil;
long loc;
{
    /* seek to the location of the record in the browse file, and */
    /* read the record */
    fseek(data_fil, loc, 0);
    if(fread((char *)prec, sizeof(struct EAMATE_W2EMPL_BRW), 1, data_fil) == 0)
        return(FALSE);
    else
        return(TRUE);
}

/*
 * display_eamate_w2employee_browse()
 *
 * This function prints a record of type EAMATE_W2EMPL_BRW.
 *
 * Input: Pointer to structure holding data record, "rec"
 *
 * Output: Data Record is Printed
 */
void display_eamate_w2employee_browse(rec)
struct EAMATE_W2EMPL_BRW *rec;
{
    printf("ssn = %s\n", rec->:ssn);
    printf("name = %s\n", rec->:name);
    printf("wages = %s\n", rec->:wages);
    printf("tips = %s\n", rec->tips);
    printf("tax = %s\n", rec->:fica_tax);
    printf("other = %s\n", rec->other);
    printf("mrn = %s\n", rec->mrn);
    printf("seq no = %s\n", rec->seq_no);
    printf("wage type = %s\n", rec->wage_type);
    printf("loc = %d\n", rec->record_loc);
    getchar();
}

/*
 * output_eamate_w2employee_browse()
 *
 * This function prints a record of type EAMATE_W2EMPL_BRW.
 *
 * Input: Pointer to structure holding data record, "rec"
 *
 * Output: Data Record is Printed
 */
output_eamate_w2employee_browse(rec, file)
struct EAMATE_W2EMPL_BRW *rec;
FILE *file;
{
    fprintf(file, "ssn = %s\n", rec->:ssn);
    fprintf(file, "name = %s\n", rec->:name);
    fprintf(file, "wages = %s\n", rec->:wages);
    fprintf(file, "tips = %s\n", rec->tips);
    fprintf(file, "tax = %s\n", rec->:fica_tax);
    fprintf(file, "other = %s\n", rec->other);
    fprintf(file, "mrn = %s\n", rec->mrn);
    fprintf(file, "seq no = %s\n", rec->seq_no);
    fprintf(file, "wage type = %s\n", rec->wage_type);
    fprintf(file, "loc = %d\n", rec->record_loc);
}

```



```

#
# Bottom Level Makefile (~/.ssapilot/src/lib/test)
#
# by Natalie Willman
#
# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependancies for the source files.
#
# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../../..
LIBDIR       = $(PROJECT_ROOT)/lib
BINDIR       = $(PROJECT_ROOT)/bin
INCDIR       = $(PROJECT_ROOT)/include

# this is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE   = libtest.a
SRC           = test_funcs.c
SRCH         = $(INCDIR)/params.h $(INCDIR)/btreetstruct.h $(INCDIR)/eamateststruct.h
LIBS         = $(LIBDIR)/libgen_eamate.a
CLIBS       = -lm -lgen_eamate
OBJ          = test_funcs.o
CFLAGS      = -I$(INCDIR) -I$(LIBDIR)
CC          = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(LIBDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(LIBDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependancies for each of the source files
depend : $(SRC)
    $(CC) -M $(CFLAGS) $(SRC) > dependlist
    sed -e '1,/^# DO NOT DELETE/!d' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv Makefile Makefile.bak

```

```

mv mm.tmp Makefile
rm -f dependlist

# directive for the executable
$(EXECUTABLE) : $(OBJ)
    ar rv $@ $(OBJ)

# directive for the executable in the binary directory
$(LIBDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(LIBDIR)

.o.a :
    ar rv $@ $(OBJ)

.c.o :
    $(CC) -c $(CFLAGS) $<

.c.a :
    $(CC) -c $(CFLAGS) $<
    ar rv $@ $(OBJ)

# DO NOT DELETE THIS LINE - make depend uses it
test_funcs.o: test_funcs.c
test_funcs.o: /usr/include/stdio.h
test_funcs.o: ../../include/params.h
test_funcs.o: ../../include/eamateststruct.h
test_funcs.o: ../../include/btreetstruct.h

```

```

/* test_funcs.c
 *
 * version 4.0
 * Natalie Willman
 * 10/26/93
 *
 * This library contains functions which are used to test
 * and debug the various components of the data conversion,
 * index application, and search applications.
 */

#include <stdio.h>
#include "params.h"
#include "eamatestruct.h"
#include "btreesstruct.h"

/* Function Prototypes
char check_data();
char check_idx();
char display_unrec();
char make_search_table();
char rw_browse();
char rw_query();
char rw_browse_off();
char rw_btrees();
char rw_detail();
char rw_dupfile();
char rw_emp_detail();
char rw_empir();
char rw_empir_idx();
char rw_error_search();
char rw_off();
char rw_parsectrl();
char rw_search_stats();
char translate_off();
char fix_parse_by_seq();
char fix_parse_all();
char change_browse();
char change_detail();

/* check_data()
 *
 * this function will check a list of given files for
 * the proper format for indexing and searching. Specifically,
 * it checks for the file to begin with an employer header record,
 * counts the employee records and makes sure that the file ends
 * with an employer final total record.
 */

char check_data(filename, out)
char filename[];
FILE *out;
{
    FILE *list_of_files, *data;
    int recount;
    char typeld, legal;
    struct EAMATE_W2EMPL_DETAIL empl;

    struct EAMATE_W2EMPLR_HEADER emplr;
    struct EAMATE_W2INTERMED_TOT inter;
    struct EAMATE_W2FINAL_TOT final;
    struct EAMATE_W2CUMEIN_TOT cum;

    /* open the list of files to check
    list_of_files = fopen(filename, "r");
    if(list_of_files == NULL)
    {
        printf("ERROR: Cannot open file %s\n", filename);
        return(ERROR);
    }

    /* while there are still files to check
    while(!feof(list_of_files))
    {
        /* read in the next file
        fscanf(list_of_files, "%s%c", filename);
        if(!feof(list_of_files))
        {
            /* open the file to check
            data = fopen(filename, "r");
            if(data == NULL)
            {
                printf("ERROR: Cannot open file %s\n", filename);
                return(ERROR);
            }

            /* set flags and counters
            legal = FALSE;
            recount = 0;
            /* While there are still records, read in the record id, and */
            /* based upon it, read in the appropriate record and check it */
            while(!feof(data))
            {
                if( fread(&typeld, sizeof(char), 1, data) != 0)
                switch(typeld)
                {
                    case MATE_W2EI: /* Employee information record
                        fread(&empl, sizeof(struct EAMATE_W2EMPL_DETAIL), 1, data);
                        /* make sure that the record is in a legal location;
                        /* after an employer header and before a final total
                        if(legal == TRUE)
                            recount++;
                        else
                            fprintf(out, "ERROR: Illegal employee record in %s\n", filename);
                        break;
                    case MATE_W2EH: /* Employer Header record
                        fprintf(out, "Employer header found for %s\n", filename);
                        fread(&emplr, sizeof(struct EAMATE_W2EMPLR_HEADER), 1, data);
                        legal = TRUE; /* employee detail records now allowed
                        break;
                    case MATE_W2IT: /* Intermediate total record
                        fread(&inter, sizeof(struct EAMATE_W2INTERMED_TOT), 1, data);
                        break;
                    case MATE_W2FT: /* Final total Record
                        fread(&final, sizeof(struct EAMATE_W2FINAL_TOT), 1, data);
                        fprintf(out, "Employer final total found for %s\n", filename);
                        legal = FALSE; /* employee detail records not allowed
                        break;
                        fprintf(out, "Record Count = %d\n", recount);
                    case MATE_W2CE: /* CUM EIN Record
                        fprintf(out, "Employer cum ein total found for %s\n", filename);

```

```

        fread(&cum, sizeof(struct ENMATE_W2CUMEIN_TOT), 1, data);
        break;
    }
    }
    fclose(data);
}

fclose(list_of_files);
return(SUCCESS);
}

/*
 * check_idx()
 *
 * this function will take the name of a matching btrec and
 * duplicate file, and will merge the two files and print a
 * human-readable version of the binary files.
 */

char check_idx(btreename, dupename, out)
char btreename[], dupename[];
FILE *out;
{
    int i, j, num;
    FILE *btreefil, *dupefil;
    struct NODE rec, *prec;
    struct RECINFO offset;

    /* open the btrec and duplicate postings files
     *
     * btreefil = fopen(btreename, "r");
     * dupefil = fopen(dupename, "r");
     * if( (btreefil == NULL) || (dupefil == NULL) )
     * {
     *     printf("ERROR: Cannot open the btrec or duplicate file\n");
     *     return(ERROR);
     * }
     *
     * set up pointers to structures
     * prec = &rec;
     *
     * While there are nodes to read
     * while(!feof(btreefil))
     * {
     *     /* Print the offset of the node in the file
     *     printf(out, "\toffset = %u\n", ftell(btreefil));
     *     /* Read the node in (if it exists) and print it for the user
     *     num = fread(prec, sizeof(struct NODE), 1, btreefil);
     *     if(num != 0)
     *     {
     *         printf(out, "\tcount = %d\n", prec->count);
     *         if(prec->leaf == TRUE)
     *         {
     *             printf(out, "\tleaf = TRUE\n");
     *             printf(out, "\tnext leaf = %d\n", prec->branch[MAXCHILD]);
     *         }
     *         else
     *         {
     *             printf(out, "\tleaf = FALSE\n");
     *             for(i = 0; i < prec->count; i++)
     *             {
     *                 char display_unrec(filename, offset, out)
     *                 char filename[], offset[];
     *                 FILE *out;
     *                 {
     *                     char trash[5188];
     *                     FILE *data;
     *                     long loc;
     *
     *                     /* open the file with the unrecognized characters
     *                     data = fopen(filename, "r");
     *                     if(data == NULL)
     *                     {
     *                         printf("ERROR: Cannot open file %s\n", filename);
     *                         return(ERROR);
     *                     }
     *
     *                     /* seek to the block before the unrecognized character - a
     *                     /* block being determined as the COM file block size - read
     *                     /* the buffer of data, and print it to the file
     *                     loc = atol(offset) - 5187;
     *                     fseek(data, loc, 0);
     *                     fgets(trash, 5187, data);
     *                     printf(out, "%s\n\n", trash);
     *
     *                     /* seek to the block containing the unrecognized character - a

```



```

/* block being determined as the COM file block size - read */
/* the buffer of data, and print it to the file */
loc = atol(offset);
fseek(data, loc, 0);
fgets(trash, 5187, data);
fprintf(out, "%s\n\n", trash);

/* seek to the block after the unrecognized character - a */
/* block being determined as the COM file block size - read */
/* the buffer of data, and print it to the file */
loc = atol(offset) + 5187;
fseek(data, loc, 0);
fgets(trash, 5187, data);
fprintf(out, "%s\n\n", trash);

fclose(data);
return(SUCCESS);
}

/*
 * rw_browse()
 *
 * this function will print a human-readable copy of the
 * browse data file specified, along with tags which
 * indicate field labels.
 */
char rw_browse(brwname, file)
char brwname[];
FILE *file;
{
    int num;
    FILE *brwfile;
    struct EAMATE_W2EMPL_BRW rec, *prec;
    brwfile = fopen(brwname, "r");
    if(brwfile == NULL)
    {
        printf("ERROR: Cannot open filename %s\n", brwname);
        return(ERROR);
    }
    prec = &rec;

    /* While there are records to read ... */
    while(!feof(brwfile))
    {
        /* Print the current location in the browse file */
        fprintf(file, "\tbrowse loc = %u\n", ftell(brwfile));
        /* Read in a record, if it exists. If it exists, print */
        /* the record. */
        num = fread(prec, sizeof(struct EAMATE_W2EMPL_BRW), 1, brwfile);
        if(num != 0)
            output_eamate_w2employee_browse(prec, file);
    }
    fclose(brwfile);
    return(SUCCESS);
}

/*
 * change_browse
 *
 * this function will modify the browse location in the
 * parsectrl file.
 */
char change_browse(ein)
char ein[];
{
    int num;
    FILE *file, *parsefile, *outfile;
    struct CTRL_FILE ctrl, *pctrl;
    char name[50], outname[50];
    char found, command[150];

    /* initialize structure pointers and flag variables */
    pctrl = &ctrl;
    found = FALSE;

    /* open the parse control file */
    strcpy(name, "parsectrl");
    strcpy(outname, "parsectrl.mod");
    parsefile = fopen(name, "r");
    if(parsefile == NULL)
    {
        printf("ERROR: Cannot open filename %s\n", name);
        return(ERROR);
    }

    /* open the parse control output file */
    outfile = fopen(outname, "w");
    if(outfile == NULL)
    {
        printf("ERROR: Cannot open filename %s\n", outname);
        return(ERROR);
    }

    /* read each record in the parse control file, and if it
     * does not match the ein which is being removed, print
     * the record to the parse control output file
     */
    while(!feof(parsefile))
    {
        num = fread(pctrl, sizeof(struct CTRL_FILE), 1, parsefile);
        if(!strcmp(pctrl->ein, ein) == 0)
        {
            printf("Enter the new browse location: \n");
            scanf("%s", pctrl->browse_loc);
            fwrite(pctrl, sizeof(struct CTRL_FILE), 1, outfile);
            found = TRUE;
        }
        else
            fwrite(pctrl, sizeof(struct CTRL_FILE), 1, outfile);
    }

    /* close the two parse control files */
    fclose(parsefile);
    fclose(outfile);

    /* if the ein was not found, then print an error

```

```

if(!found)
{
    printf("ERROR: No entry in parsectrl for %s\n", ein);
    return(ERROR);
}
else /* otherwise, ... */
{
    /* overwrite the old parse control file with the new file */
    sprintf(command, "mv -f %s %s", outname, name);
    printf("%s modified ...\n", name);
    printf("The browse file for this EIN must be moved to the new location\n");
    system(command);
}

/* fix_parse_all
*
* this function will fix a parse run by removing all reference
* to the specified ein in the control files.
*/

char fix_parse_all(ein)
{
    int num;
    FILE *file, *parsefile, *outfile;
    struct EAMATE_W2EMPLR_INFO rec_info, *prec_info;
    struct CTRL_FILE ctrl, *pctrl;
    char name[50], outname[50];
    char found, command[150];

    /* Initialize structure pointers and flag variables */
    prec_info = &rec_info;
    pctrl = &ctrl;
    found = FALSE;

    /* open the parse control file
    strcpy(name, "parsectrl");
    strcpy(outname, "parsectrl.mod");
    parsefile = fopen(name, "r");
    if(parsefile == NULL)
    {
        printf("ERROR: Cannot open filename %s\n", name);
        return(ERROR);
    }

    /* open the parse control output file
    outfile = fopen(outname, "w");
    if(outfile == NULL)
    {
        printf("ERROR: Cannot open filename %s\n", outname);
        return(ERROR);
    }

    /* read each record in the parse control file, and if it
    /* does not match the ein which is being removed, print
    /* the record to the parse control output file
    while(!feof(parsefile))
    {
        num = fread(pctrl, sizeof(struct CTRL_FILE), 1, parsefile);
        if(strcmp(pctrl->ein, ein) == 0)
            found = TRUE;
        else
            fwrite(pctrl, sizeof(struct CTRL_FILE), 1, outfile);
    }

    /* close the two parse control files
    fclose(parsefile);
    fclose(outfile);

    /* if the ein was not found, then print an error
    if(!found)
    {
        printf("ERROR: No entry in parsectrl for %s\n", ein);
        return(ERROR);
    }
    else /* otherwise, ...
    {
        /* overwrite the old parse control file with the new file */
        sprintf(command, "mv -f %s %s", outname, name);
        printf("%s modified ...\n", name);
        printf("The browse file for this EIN must be moved to the new location\n");
        system(command);
    }

    /* open the employer header info files (input & output)
    strcpy(name, "1991-employers.text");
    strcpy(outname, "1991-employers.text.mod");
    file = fopen(name, "r");
    if(file == NULL)
    {
        printf("ERROR: Cannot open filename %s\n", name);
        return(ERROR);
    }

    outfile = fopen(outname, "w");
    if(outfile == NULL)
    {
        printf("ERROR: Cannot open filename %s\n", outname);
        return(ERROR);
    }

    /* read in each of the employer header info records, and if */
    /* the ein does not match the ein we are erasing, print the */
    /* record to the output file
    while(!feof(file))
    {
        num = fread(prec_info, sizeof(struct EAMATE_W2EMPLR_INFO), 1, file);
        if (num != 0)
        {
            if(strcmp(ein, prec_info->ein) != 0)
                fwrite(prec_info, sizeof(struct EAMATE_W2EMPLR_INFO), 1, outfile);
        }
    }

    fclose(file);
    fclose(outfile);

    /* overwrite the old employer header file with the new updated one */
    sprintf(command, "mv -f %s %s", outname, name);
    printf("%s modified ...\n", name);
    system(command);
}

```

```

/* User must delete the detail and browse files for the ein (safety check!) */
printf("Detail and Browse files for this EIN needs to be removed - check einstats file
for location\n");
return(SUCCESS);
}

/*
 * fix_parse_by_seq
 *
 * this function will fix a parse run by removing all reference
 * to the specified ein/sequence number in the control and
 * browse data files.
 */
char fix_parse_by_seq(ein, seq)
char ein[], seq[];
{
    int num;
    FILE *file, *parsefile, *outfile;
    struct EAMATE_W2EMPL_BRW rec, *prec;
    struct EAMATE_W2EMPLR_INFO rec_info, *prec_info;
    struct CTRL_FILE ctrl, *pctrl;
    char name[50], outname[50];
    char found, command[150];

    /* Initialize structure pointers and flag variables */
    prec = &rec;
    prec_info = &rec_info;
    pctrl = &ctrl;
    found = FALSE;

    /* check the parse control file to make sure the given ein
    /* exists
    parsefile = fopen("parsectrl", "r");
    if(parsefile == NULL)
    {
        printf("ERROR: Cannot open filename parsectrl\n");
        return(ERROR);
    }
    while( !feof(parsefile) && (!found) )
    {
        num = fread(pctrl, sizeof(struct CTRL_FILE), 1, parsefile);
        if(strcmp(pctrl->ein, ein) == 0)
            found = TRUE;
    }

    /* if this ein does not exist, print an error
    if(!found)
    {
        printf("ERROR: No entry in parsectrl for %s\n", ein);
        return(ERROR);
    }
    fclose(parsefile);

    /* get the browse filename, and open the file, as well as
    /* a temporary output file
    cr_browse_filename(name, "1991", pctrl->browse_loc, ein);
    file = fopen(name, "r");
    if(file == NULL)
    {
        printf("ERROR: Cannot open filename %s\n", name);
        return(ERROR);
    }
    strcpy(outname, name);
    strcat(outname, ".mod");
    outfile = fopen(outname, "w");
    if(outfile == NULL)
    {
        printf("ERROR: Cannot open filename %s\n", outname);
        return(ERROR);
    }

    /* read in each of the records, and check to see if the seq
    /* number of the record matches the one that is being
    /* removed. If not, then print the record to the output
    while(!feof(file))
    {
        num = fread(prec, sizeof(struct EAMATE_W2EMPL_BRW), 1, file);
        if( (num != 0) && (strcmp(seq, prec->seq_no) != 0) )
            fwrite(prec, sizeof(struct EAMATE_W2EMPL_BRW), 1, outfile);
    }
    fclose(file);
    fclose(outfile);

    /* replace the old browse file with the new modified file
    sprintf(command, "mv -f %s %s", outname, name);
    system(command);
    printf("%s modified ...\n", name);

    /* open the employer header information file, and an output
    /* file for it
    strcpy(name, "1991.employers.text");
    strcpy(outname, "1991.employers.text.mod");
    file = fopen(name, "r");
    if(file == NULL)
    {
        printf("ERROR: Cannot open filename %s\n", name);
        return(ERROR);
    }
    outfile = fopen(outname, "w");
    if(outfile == NULL)
    {
        printf("ERROR: Cannot open filename %s\n", outname);
        return(ERROR);
    }

    /* read each of the records, and check to see if it is for the
    /* ein and sequence number being removed. If not, print the record
    /* to the output file
    while(!feof(file))
    {
        num = fread(prec_info, sizeof(struct EAMATE_W2EMPLR_INFO), 1, file);
        if (num != 0)
        {
            if(strcmp(ein, prec_info->ein) != 0)
                fwrite(prec_info, sizeof(struct EAMATE_W2EMPLR_INFO), 1, outfile);
        }
    }
}

```



```

    else if(strcmp(seq, prec_info->seq_no) != 0)
        fwrite(prec_info, sizeof(struct EAMATE_W2EMPLR_INFO), 1, outfile);
    }
}

fclose(outfile);
fclose(outfile);

/* overwrite the old employer header file
   sprintf(command, "mv -f %s %s", outname, name);
   system(command);
   printf("%s modified ...\\n", name);
*/

/* User must delete the detail file - (safety check)
   printf("Detail file for this EIN/SEQ needs to be removed - check e1stats file for loc
   action\\n");
   return(SUCCESS);
}

/*
 * rw_query()
 *
 * this function will print a human-readable copy of the
 * query data file specified, along with tags which
 * indicate field labels.
 */

char rw_query(qname, file)
char qname[];
FILE *file;
{
    int num;
    FILE *qfile;
    struct USER_QUERY rec, *prec;

    qfile = fopen(qname, "r");
    if(qfile == NULL)
    {
        printf("ERROR: Cannot open file %s\\n", qname);
        return(ERROR);
    }

    prec = &rec;

    /* While there are records to read ...
       while(!feof(qfile))
    {
        /* Print the current location in the browse file
        fprintf(file, "\\trowseloc = %u\\n", ftell(brwfile));
        /* Read in a record, if it exists. If it exists, print
        /* the record.
        num = fread(prec, sizeof(struct EAMATE_W2EMPLR_BRW), 1, brwfile);
        if(num != 0)
            output_eamate_w2employee_browse(prec, file);
        }
        fclose(brwfile);
        return(SUCCESS);
    }

    /*
    * rw_btree()
    *
    * this function will print a human-readable copy of the
    * btree data file specified, along with tags which
    * indicate field labels.
    */

    char rw_btree(btreename, out)
    char btreename[];
    FILE *out;
    {
        fclose(qfile);
        return(SUCCESS);
    }
}

```

```

int i, num;
FILE *btreefil;
struct NODE rec, *prec;

btreefil = fopen(btreename, "r");
if (btreefil == NULL)
{
    printf("ERROR: Cannot open file %s\n", btreename);
    return(ERROR);
}

prec = &rec;

/* While there are nodes to read
while(!feof(btreefil))
{
    /* Print the offset of the node in the file
    fprintf(out, "\toffset = %u\n", ftell(btreefil));
    /* Read the node in (if it exists) and print it for the user
    num = fread(prec, sizeof(struct NODE), 1, btreefil);
    if (num != 0)
    {
        fprintf(out, "\tcount = %d\n", prec->count);
        if (prec->leaf == TRUE)
        {
            fprintf(out, "\tleaf = TRUE\n");
            fprintf(out, "\tnext leaf = %d\n", prec->branch[MAXCHILD]);
        }
        else
        {
            fprintf(out, "\tleaf = FALSE\n");
            for (i = 0; i < prec->count; i++)
            {
                fprintf(out, "\tkey %d = %s\n", i, prec->key[i]);
                fprintf(out, "\tfrequency %d = %d\n", i, prec->freq[i]);
                fprintf(out, "\tbranch %d = %d\n", i, prec->branch[i]);
            }
            if (prec->leaf == FALSE)
            {
                fprintf(out, "\tbranch %d = %d\n", i, prec->branch[i]);
                fprintf(out, "\n\n");
            }
        }
        fclose(btreefil);
        return(SUCCESS);
    }
}

/* rw_detail()
*
* This function will print a human-readable copy of the
* detail data file specified, along with tags which
* indicate field labels.
*/

char rw_detail(detailname, file)
char detailname[];
FILE *file;
{
    FILE *detailfile;
    char typeid;
    struct EMATE_W2EMPL_DETAIL empl;
    struct EMATE_W2EMPLR_HEADER emplr;
    struct EMATE_W2INTERMED_TOT inter;
    struct EMATE_W2FINAL_TOT final;
    struct EMATE_W2CUMEIN_TOT cumein;

    /* open the file of records to read
    detailfile = fopen(detailname, "r");
    if (detailfile == NULL)
    {
        printf("ERROR: Cannot open file %s\n", detailname);
        return(ERROR);
    }

    /* While there are still records, read in the record id, and
    /* based upon it, read in the appropriate record and print it
    /* as well as the offset to the record in the data file
    while(!feof(detailfile))
    {
        fprintf(file, "offset = %d\n", ftell(detailfile));
        if (fread(&typeid, sizeof(char), 1, detailfile) != 0)
            switch(typeid)
            {
                case MATE_W2E1: /* Employee information record
                    fread(&empl, sizeof(struct EMATE_W2EMPL_DETAIL), 1, detailfile);
                    output_imate_w2employee_detail(empl, file);
                    break;
                case MATE_W2E2: /* Employer header record
                    fread(&emplr, sizeof(struct EMATE_W2EMPLR_HEADER), 1, detailfile);
                    output_imate_w2header(emplr, file);
                    break;
                case MATE_W2IT: /* Intermediate total record
                    fread(&inter, sizeof(struct EMATE_W2INTERMED_TOT), 1, detailfile);
                    output_imate_w2inter_tot(inter, file);
                    break;
                case MATE_W2FT: /* Final total record
                    fread(&final, sizeof(struct EMATE_W2FINAL_TOT), 1, detailfile);
                    output_imate_w2final_tot(final, file);
                    break;
                case MATE_W2CE: /* cumulative ein total record
                    fread(&cumein, sizeof(struct EMATE_W2CUMEIN_TOT), 1, detailfile);
                    output_imate_w2cumein_tot(cumein, file);
                    break;
            }
        }
        fclose(detailfile);
        return(SUCCESS);
    }

    /* rw_dupefile()
    *
    * This function will print a human-readable copy of the
    * duplicate postings file specified, along with tags which
    * indicate field labels.
    */

    char rw_dupfile(dupename, out)
    char dupename[];
    FILE *out;
    {
        FILE *dupefile;
        struct RECINFO offset;

```

```

/* Open the file
dupefile = fopen(dupename, "r");
if (dupefile == NULL)
{
    printf("ERROR: Cannot open file %s\n", dupename);
    return(ERROR);
}

/* While there are duplicate records, print the offset of
/* the data, read in the count of duplicates, and read in */
/* the list of offsets and print them
while (!feof(dupefile))
{
    fprintf(out, "\toffset = %d ", ftell(dupefile));
    fread(&offset, sizeof(struct RECINFO), 1, dupefile);
    if (!feof(dupefile))
    {
        fprintf(out, "\toffset = %d\n", offset.dupe_offset);
        fprintf(out, "\tweight = %d\n", offset.dupe_weight);
    }
}

fclose(dupefile);
return(SUCCESS);
}

/*
 * rw_emp_detail()
 *
 * this function will print a human-readable copy of the
 * employee detail file specified, along with tags which
 * indicate field labels.
 */
char rw_emp_detail(detailname, file)
char detailname[];
FILE *file;
{
    FILE *detailfile;
    struct EAMATE_W2EMPLR_DETAIL empl;

    /* open the file of records to read
    detailfile = fopen(detailname, "r");
    if (detailfile == NULL)
    {
        printf("ERROR: Cannot open file %s\n", detailname);
        return(ERROR);
    }

    while (!feof(detailfile))
    {
        fread(&empl, sizeof(struct EAMATE_W2EMPLR_DETAIL), 1, detailfile);
        if (!feof(detailfile))
            output_eamate_w2employee_detail(empl, file);
    }

    fclose(detailfile);
    return(SUCCESS);
}

/*
 * rw_emplr()
 *
 * this function will print a human-readable copy of the
 * employer header file specified, along with tags which
 * indicate field labels.
 */
char rw_emplr(emplname, file)
char emplname[];
FILE *file;
{
    int num;
    FILE *emplrfile;
    struct EAMATE_W2EMPLR_INFO rec, *prec;

    /* Open the file from which to read
    emplrfile = fopen(emplname, "r");
    if (emplrfile == NULL)
    {
        printf("ERROR: Cannot open file %s\n", emplname);
        return(ERROR);
    }

    /* Initialize record pointer
    prec = &rec;

    /* While there are records, print the offset of the record */
    /* and read and print the record
    while (!feof(emplrfile))
    {
        fprintf(file, "\templyrloc = %u\n", ftell(emplrfile));
        num = fread(prec, sizeof(struct EAMATE_W2EMPLR_INFO), 1, emplrfile);
        if (num != 0)
            output_eamate_w2header_info(prec, file);
    }

    fclose(emplrfile);
    return(SUCCESS);
}

/*
 * rw_emplr_idx()
 *
 * this function will print a human-readable copy of the
 * employer header index file specified, along with tags which
 * indicate field labels.
 */
char rw_emplr_idx(idxname, out)
char idxname[];
FILE *out;
{
    int i, num;
    FILE *idxfile;
    struct EMPHR_IDX rec, *prec;

    idxfile = fopen(idxname, "r");
    if (idxfile == NULL)
    {
        printf("ERROR: Cannot open file %s\n", idxname);
    }
}

```



```

return(ERROR);
}

prec = &rec;

/* While there are nodes to read
while(!feof(idxfile))
{
    /* Print the offset of the node in the file
    fprintf(out, "\toffset = %u\n", ftell(idxfile));
    /* Read the node in (if it exists) and print it for the user
    num = fread(prec, sizeof(struct EMPLR_IDX), 1, idxfile);
    if(num != 0)
    {
        fprintf(out, "\tein = %s\n", prec->ein);
        fprintf(out, "\toffset %u\n", prec->offset);
        fprintf(out, "\n\n");
    }
    fclose(idxfile);
    return(SUCCESS);
}

/*
* rw_error_search()
*
* This function will print a human-readable copy of any
* records in the search statistics file which contain an error
* flag, along with tags which indicate field labels.
*
*/

char rw_error_search(searchname, out)
char searchname[];
FILE *out;
{
    int num;
    FILE *searchfile;
    struct TEST_DATA rec, *prec;

    searchfile = fopen(searchname, "r");
    if(searchfile == NULL)
    {
        printf("ERROR: Cannot open file %s\n", searchname);
        return(ERROR);
    }

    prec = &rec;
    /* While there are records to read ...
    while(!feof(searchfile))
    {
        /* Read in a record, if it exists, print the record.
        num = fread(prec, sizeof(struct TEST_DATA), 1, searchfile);
        if(num != 0)
            if(prec->nummatch == -1)
            {
                printf(out, "user = %d\n", prec->user);
                printf(out, "year = %s\n", prec->year);
                printf(out, "ein = %s\n", prec->ein);
                printf(out, "first = %s\n", prec->first);
                printf(out, "last = %s\n", prec->last);
                printf(out, "searchname = %s\n", prec->searchname);
                printf(out, "ssn = %s\n", prec->ssn);
            }
        }
    }

    /* num grams = %d\n", prec->numgrams);
    fprintf(out, "num matches = %d\n", prec->nummatch);
    fprintf(out, "recs pruned = %d\n", prec->recspruned);
    fprintf(out, "elapsed time = %d\n", prec->etime);
    fprintf(out, "write time = %d\n", prec->wtime);
    fprintf(out, "memory allocated = %d\n", prec->mem);
    fprintf(out, "cpu time = %f\n", prec->ctime);
}

fclose(searchfile);
return(SUCCESS);
}

/*
* rw_off()
*
* This function will print a human-readable copy of the
* list of offsets file specified, along with tags which
* indicate field labels.
*
*/

char rw_off(offname, out)
char offname[];
FILE *out;
{
    FILE *off_file;
    long offset;

    /* open the file of records to read
    off_file = fopen(offname, "r");
    if(off_file == NULL)
    {
        printf("ERROR: Cannot open file %s\n", offname);
        return(ERROR);
    }

    /* While there are still records, read in the record id, and
    /* based upon it, read in the appropriate record and print it
    /* as well as the offset to the record in the data file
    while(!feof(off_file))
    {
        fread(&offset, sizeof(long), 1, off_file);
        if(!feof(off_file))
            fprintf(out, "offset = %d\n", offset);
    }

    fclose(off_file);
    return(SUCCESS);
}

/*
* rw_parsectrl()
*
* This function will print a human-readable copy of the
* parse control file specified, along with tags which
* indicate field labels.
*
*/

char rw_parsectrl(ctrlname, out)
char ctrlname[];

```

```

FILE *out;
{
    int num;
    FILE *ctrlfil;
    struct CTRL_FILE rec, *prec;

    ctrlfil = fopen(ctrlname, "r");
    if(ctrlfil == NULL)
    {
        printf("ERROR: Cannot open file %s\n", ctrlname);
        return(ERROR);
    }

    prec = &rec;
    /* While there are records to read ...
    while(!feof(ctrlfil))
    {
        /* Read in a record, if it exists. If it exists, print */
        /* the record.
        printf(out, "offset = %d\n", ftell(ctrlfil));
        num = fread(prec, sizeof(struct CTRL_FILE), 1, ctrlfil);
        if(num != 0)
        {
            printf(out, "user = %d\n", prec->user);
            printf(out, "year = %s\n", prec->year);
            printf(out, "ein = %s\n", prec->ein);
            printf(out, "first = %s\n", prec->first);
            printf(out, "last = %s\n", prec->last);
            printf(out, "searchname = %s\n", prec->searchname);
            printf(out, "ssn = %s\n", prec->ssn);
            printf(out, "num grams = %d\n", prec->numgrams);
            printf(out, "num matches = %d\n", prec->nummatch);
            printf(out, "recs pruned = %d\n", prec->recspruned);
            printf(out, "elapsed time = %d\n", prec->etime);
            printf(out, "write time = %d\n", prec->wtime);
            printf(out, "memory allocated = %d\n", prec->mem);
            printf(out, "cpu time = %f\n", prec->ctime);
        }
        fclose(ctrlfil);
        return(SUCCESS);
    }
}

/* make_search_table()
*
* this function will print a human-readable copy of the
* search statistics file specified, in a table format.
*
*/

char make_search_table(statsname, out)
char statsname[];
FILE *out;
{
    int num;
    FILE *statsfil;
    struct TEST_DATA rec, *prec;

    statsfil = fopen(statsname, "r");
    if(statsfil == NULL)
    {
        printf("ERROR: Cannot open file %s\n", statsname);
        return(ERROR);
    }

    prec = &rec;

    printf(out, "%11s%3s%10s%10s%7s%10s%7s%7s%13s%15s%27s\n",
        "EIN", "GR", "CP", "MATCHES", "PRUNED", "LEFT",
        "MEMORY", "ELPSD", "CPU", "WRITE", "FIRST", "LAST", "SEARCH");

    /* While there are records to read ...
    while(!feof(statsfil))
    {
        /* Read in a record, if it exists. If it exists, print */
        /* the record.
        num = fread(prec, sizeof(struct TEST_DATA), 1, statsfil);
        if(num != 0)
            printf(out, "%11s%3d%10d%10d%7d%10d%7d%7d%13s%15s%27s\n",

```

```

    prec->eln, prec->numgrams, prec->grprune,
    prec->nummatch, prec->recspruned,
    prec->nummatch - prec->recspruned,
    prec->mem, prec->etime, prec->ctime, prec->wtline,
    prec->first, prec->last, prec->searchname);
}
fclose(statsfil);
return(SUCCESS);
}

/*
 * translate_off()
 *
 * this function will print a human-readable copy of the
 * browse data records specified by the given list of offsets
 * file, with a label describing the field contents.
 */

char translate_off(offname, brwname, file)
char offname[], brwname[];
FILE *file;
{
    int num;
    FILE *off_fil, *brw_file;
    struct EAMATE_W2EMPL_BRW rec, *prec;
    long offset;

    /* open the file of records to read */
    off_fil = fopen(offname, "r"); /* offset file */
    brw_file = fopen(brwname, "r"); /* browse record file */
    if (!off_fil || !brw_file)
    {
        printf("ERROR: Cannot open file %s or %s\n", offname, brwname);
        return(ERROR);
    }

    prec = &rec;

    /* While there are still records, read in the record id, and */
    /* based upon it, read in the appropriate record and print it */
    /* as well as the offset to the record in the data file */
    while (!feof(off_fil))
    {
        fread(&offset, sizeof(long), 1, off_fil);
        if (!feof(off_fil))
        {
            fseek(brw_file, offset, 0);
            num = fread(prec, sizeof(struct EAMATE_W2EMPL_BRW), 1, brw_file);
            if (num != 0)
            {
                output_eamate_w2employee_browse(prec, file);
            }
        }
        fclose(brw_file);
        fclose(off_fil);
        return(SUCCESS);
    }
}

```



```

# Bottom Level Makefile (~/.ssapilot/src/blm/client)
#
# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependencies for the source files.
#
# This is a list of the key directories in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE = client
SRC         = client.c
LIBS        = $(LIBDIR)/libgen_eamate.a
CLIBS       = -lgen_eamate
OBJ         = client.o
CFLAGS      = -I$(INCDIR) -L$(LIBDIR)
CC          = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependencies for each of the source files
depend : $(SRC)
    sed -e '1,/^# DO NOT DELETE/Id' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv mm.tmp Makefile.bak
    mv mm.tmp Makefile
    rm -f dependlist

```

```

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
    $(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(BINDIR)

.c.o :
    $(CC) -c $(CFLAGS) <<

# DO NOT DELETE THIS LINE - make depend uses it
client.o : client.c
client.o : /usr/include/stdio.h
client.o : /usr/include/stdlib.h
client.o : /usr/include/sys/types.h
client.o : /usr/include/sys/select.h
client.o : /usr/include/sys/time.h
client.o : /usr/include/sys/types.h
client.o : /usr/include/time.h
client.o : /usr/include/sys/sysmacros.h
client.o : /usr/include/sys/time.h
client.o : /usr/include/malloc.h
client.o : /usr/include/math.h
client.o : /usr/include/floatpoint.h
client.o : /usr/include/sys/ieeefp.h
client.o : /usr/include/string.h
client.o : /usr/include/rpc/rpc.h
client.o : /usr/include/rpc/types.h
client.o : /usr/include/sys/types.h
client.o : /usr/include/sys/time.h
client.o : /usr/include/tuser.h
client.o : /usr/include/sys/tuser.h
client.o : /usr/include/fcntl.h
client.o : /usr/include/memory.h
client.o : /usr/include/rpc/xdr.h
client.o : /usr/include/sys/byteorder.h
client.o : /usr/include/sys/types.h
client.o : /usr/include/stdio.h
client.o : /usr/include/rpc/auth.h
client.o : /usr/include/rpc/xdr.h
client.o : /usr/include/sys/cred.h
client.o : /usr/include/sys/t_lock.h
client.o : /usr/include/sys/machlock.h
client.o : /usr/include/sys/types.h
client.o : /usr/include/sys/dk1_kinfo.h
client.o : /usr/include/sys/types.h
client.o : /usr/include/sys/dl.h
client.o : /usr/include/sys/sleepq.h
client.o : /usr/include/sys/turnstile.h
client.o : /usr/include/sys/types.h
client.o : /usr/include/sys/param.h
client.o : /usr/include/limits.h
client.o : /usr/include/unistd.h
client.o : /usr/include/sys/fcntl.h
client.o : /usr/include/sys/signal.h
client.o : /usr/include/vm/faultcode.h
client.o : /usr/include/sys/types.h
client.o : /usr/include/sys/pirec.h
client.o : /usr/include/sys/sleepq.h
client.o : /usr/include/sys/mutex.h
client.o : /usr/include/sys/types.h
client.o : /usr/include/sys/machlock.h
client.o : /usr/include/sys/turnstile.h

```

```
client.o: /usr/include/sys/dk1_kinfo.h
client.o: /usr/include/rpc/clnt.h
client.o: /usr/include/rpc/rpc_com.h
client.o: /usr/include/sys/netconfig.h
client.o: /usr/include/rpc/rpc_msg.h
client.o: /usr/include/rpc/clnt.h
client.o: /usr/include/rpc/auth_sys.h
client.o: /usr/include/rpc/auth_des.h
client.o: /usr/include/rpc/auth_kerb.h
client.o: /usr/include/kerberos/krb.h
client.o: /usr/include/kerberos/mit-copyright.h
client.o: /usr/include/kerberos/des.h
client.o: /usr/include/kerberos/mit-copyright.h
client.o: /usr/include/sys/socket.h
client.o: /usr/include/sys/netconfig.h
client.o: /usr/include/netinet/in.h
client.o: /usr/include/sys/steam.h
client.o: /usr/include/sys/vnode.h
client.o: /usr/include/sys/types.h
client.o: /usr/include/sys/t_lock.h
client.o: /usr/include/sys/time.h
client.o: /usr/include/sys/cred.h
client.o: /usr/include/sys/uio.h
client.o: /usr/include/sys/types.h
client.o: /usr/include/sys/poll.h
client.o: /usr/include/sys/rndep.h
client.o: /usr/include/sys/cred.h
client.o: /usr/include/sys/t_lock.h
client.o: /usr/include/sys/byteorder.h
client.o: /usr/include/rpc/svc.h
client.o: /usr/include/rpc/rpc_com.h
client.o: /usr/include/rpc/rpc_msg.h
client.o: /usr/include/rpc/svc_auth.h
client.o: /usr/include/rpc/svc.h
client.o: /usr/include/rpc/rpcb_clnt.h
client.o: /usr/include/rpc/types.h
client.o: /usr/include/rpc/rpcb_prot.h
client.o: /usr/include/rpc/rpc.h
client.o: ../../include/params.h
client.o: ../../include/eamateststruct.h
client.o: ../../include/btreestruct.h
```

```

/*
 * client.c
 * version 4.0
 * 11/2/93
 *
 * by Natalie Willman
 *
 * This program is a small client program to be run on the
 * file server to exercise the search engine for debugging.
 */

/* Include files */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/time.h>
#include <malloc.h>
#include <math.h>
#include <string.h>
#include <rpc/rpc.h>
#include "params.h"
#include "samestruct.h" /* Data file structure definitions */
#include "btstruct.h" /* B+ tree structure definitions */

#define SERVER "demeter"

/*
 * MAIN PROGRAM
 *
 * This program presents a list of options to the user and will make
 * the appropriate rpc call based upon the user selection. It will
 * take information from the user to be passed to the rpc call, and will
 * print results on the screen.
 */

main(argc, argv)
int argc;
char *argv[];
{
    char temp[51];
    char selection;
    int user_no;
    struct USER_QUERY user_query; /* query structure for server info */
    int res, x, count; /* counter and result variables */
    bool t_resl; /* result variable */
    char filename[50], namefile[50]; /* filename strings */
    char namestring[100]; /* filename string */
    FILE *qfile, *nfile; /* file pointers - query/name file */
    CLIENT *clnt; /* client handle pointer */
    struct timeval timeout; /* timeout data structure */

    /* If the user has only typed "client" at the command line, print */
    /* a list of options and retrieve their selection */
    if(argc < 2)
    {
        printf("1 Employer Header Info -- All\n");
        printf("2 Employer Header Info -- Seq\n");
        printf("3 Single Query\n");
        printf("4 Detail Record\n");
        printf("5 Browse Report\n");
        printf("6 Blanket Info\n");
        printf("7 Print Report\n");
    }

    printf("8 Add Matches\n");
    printf("\n\n");
    printf("Enter the Selection: ");
    scanf("%c", &selection);
}

/* Otherwise, they have entered a name file, so hardcode the */
/* choice to single query, and retrieve the name file */
else
{
    selection = '3';
    strcpy(namefile, argv[1]);
}

/* Switch based upon the user's selection of query request */
switch(selection)
{
    /* Employer Header Information Request -- All headers for EIN */
    case '1':
        printf("Employer Header Information - All\n");

        /* Get the year, ein and user number from the user and */
        /* fill in the rest of the parameters with null values */
        printf("Enter the year: ");
        gets(user_query.year);

        printf("Enter the EIN: ");
        gets(user_query.ein);

        printf("Enter the User No: ");
        gets(temp);
        user_no = atoi(temp);

        user_query.res[0] = 0;
        user_query.seq_no[0] = 0;
        user_query.first[0] = 0;
        user_query.last[0] = 0;
        user_query.ssn[0] = 0;
        user_query.offset[0] = 0;

        /* print the parameters on the screen for debugging purposes */
        printf("%s\n%s\n", user_query.year, user_query.ein, user_no);

        /* Open the user query file */
        sprintf(qfilename, "query%d.txt", user_no);
        qfile = fopen(qfilename, "wb");
        if(qfile == NULL)
        {
            printf("ERROR: Cannot open file %s\n", qfilename);
            exit(1);
        }

        /* Write the query structure to the file for the server */
        if(fwrite(&user_query, sizeof(struct USER_QUERY), 1, qfile) == 0)
        {
            printf("ERROR: Cannot write query\n");
            fclose(qfile);
            exit(1);
        }
        fclose(qfile);

        /* Call the rpc function, and print the result */
        res = callrpc(SERVER, 0x30000000, 1, 1, xdr_int, &user_no, xdr_int, &x);
        if(res)
            clnt_perrno(res);
}

```



```

else
    printf("%d\n", x);
break;

/* Employer Header Information Request -- Specific Sequence No. */
case '2':
    printf("Employer Header Information -- Seq\n");

/* Get the year, ein, sequence # and user # from the user */
/* and fill in the rest of the parameters with null values */
printf("Enter the year: ");
gets(user_query.year);

printf("Enter the EIN: ");
gets(user_query.ein);

printf("Enter the seq no: ");
gets(user_query.seq_no);

printf("Enter the User No: ");
gets(temp);
user_no = atoi(temp);

user_query.est[0] = 0;
user_query.first[0] = 0;
user_query.last[0] = 0;
user_query.ssn[0] = 0;
user_query.offset[0] = 0;

/* print the parameters on the screen for debugging purposes */
printf("%s\n%s\n%s\n%s\n", user_query.year, user_query.ein,
        user_query.seq_no, user_no);

/* Open the user query file
sprintf(qfilename, "query%d.txt", user_no);
qfile = fopen(qfilename, "wb");
if(qfile == NULL)
{
    printf("ERROR: Cannot open file %s\n", qfilename);
    exit(1);
}

/* Write the query structure to the file for the server */
if(fwrite(&user_query, sizeof(struct USER_QUERY), 1, qfile) == 0)
{
    printf("ERROR: Cannot write query\n");
    fclose(qfile);
    exit(1);
}
fclose(qfile);

/* Call the rpc function, and print the result
res = callrpc(SERVER, 0x36000000, 1, 1, xdr_int, &user_no, xdr_int, &x);
if(res)
    cint_perrno(res);
else
    printf("%d\n", x);
break;

/* Single Query Selection
case '3':
    printf("Single Query selection\n");
/* Get the year, ein and user number from the user. If the
/* user has not entered a name file, get first/last name and

```

```

/* SSN. Fill in the rest of the parameters with null values */
printf("Enter the year: ");
gets(user_query.year);

printf("Enter the EIN: ");
gets(user_query.ein);

printf("Enter the User No: ");
gets(temp);
user_no = atoi(temp);

user_query.est[0] = 0;
user_query.seq_no[0] = 0;
user_query.offset[0] = 0;

/* If no name file was entered ... get the name from user
if(argc < 2)
{
    printf("Enter the First Name: ");
    gets(user_query.first);

    printf("Enter the Last Name: ");
    gets(user_query.last);

    printf("Enter the SSN: ");
    gets(user_query.ssn);

/* print the parameters on the screen for debugging purposes */
printf("%s\n%s\n%s\n%s\n",
        user_query.year, user_query.ein, user_query.first,
        user_query.last, user_query.ssn, user_no);

/* Open the user query file
sprintf(qfilename, "query%d.txt", user_no);
qfile = fopen(qfilename, "wb");
if(qfile == NULL)
{
    printf("ERROR: Cannot open file %s\n", qfilename);
    exit(1);
}

/* Write the query structure to the file for the server */
if(fwrite(&user_query, sizeof(struct USER_QUERY), 1, qfile) == 0)
{
    printf("ERROR: Cannot write query\n");
    fclose(qfile);
    exit(1);
}
fclose(qfile);

/* Create a Client Handle
cint = cint_create(SERVER, 0x31000000, 1, "udp");
if(cint == NULL)
{
    printf("ERROR: Cannot create client handle\n");
    exit(1);
}

/* Modify the timeout value to control retransmissions
timeout.tv_sec = 150;
timeout.tv_usec = 0;
res1 = cint_control(cint, CLSET_RETRY_TIMEOUT, &timeout);
if(res1 == FALSE)
    printf("ERROR: Cannot modify client handle\n");

```

```

/* Set the overall timeout, Call the rpc function, and print */
/* the result */
timeout.tv_sec = 150;
timeout.tv_usec = 0;
res = c_int_call(cint,1,xdr_int,&user_no,xdr_int,&x, timeout);
if(res)
else
    cint_perrno(res);
    printf("%d\n", x);

/* destroy the client handle */
cint_destroy(cint);

/* If a name file was entered ... get the name from the file */
else
{
    /* Open the name file */
    nfile = fopen(namefile, "r");
    if(nfile == NULL)
    {
        printf("ERROR: Cannot open file %s\n", namefile);
        exit(1);
    }

    /* While there are still names in the file ... */
    count = 0;
    while(!feof(nfile))
    {
        /* get the name from the file */
        fgets(namestring, 100, nfile);
        if(!feof(nfile))
        {
            /* keep track of the number of names, and parse the string */
            /* into first and last names */
            count++;
            namestring[strlen(namestring)-1] = 0;
            parse_query(namestring, &user_query);

            /* print the information for debugging purposes */
            printf("%s\n%s\n%s\n%s\n%s\n",
                user_query.year, user_query.eln, user_query.first,
                user_query.last, user_query.ssn, user_no, count);

            /* open the user query file */
            sprintf(qfilename, "query%d.txt", user_no);
            qfile = fopen(qfilename, "wb");
            if(qfile == NULL)
            {
                printf("ERROR: Cannot open file %s\n", qfilename);
                exit(1);
            }

            /* write the user query structure */
            if(!fwrite(&user_query, sizeof(struct USER_QUERY), 1, qfile) == 0)
            {
                printf("ERROR: Cannot write query\n");
                fclose(qfile);
                exit(1);
            }
            fclose(qfile);
        }
    }
}

/* create the client handle */
cint = c_int_create(SERVER,0x31000000,1,"udp");
if(cint == NULL)
{
    printf("ERROR: Cannot create client handle\n");
    exit(1);
}

/* set the retransmission timeout for the handle */
timeout.tv_sec = 30;
timeout.tv_usec = 0;
res1 = c_int_control(cint, C1SET_RETRY_TIMEOUT, &timeout);
if(res1 == FALSE)
    printf("ERROR: Cannot modify client handle\n");

/* set the overall timeout and call the rpc function and */
/* print the result */
timeout.tv_sec = 30;
timeout.tv_usec = 0;
res = c_int_call(cint,1,xdr_int,&user_no,xdr_int,&x, timeout);
if(res)
    cint_perrno(res);
else
    printf("%d\n", x);

/* destroy the client handle */
cint_destroy(cint);

/* close the name file */
fclose(nfile);
break;

/* Get the employee detail information */
case '4':
    printf("Employee Detail Information\n");

    /* Get the year, eln, sequence number, detail offset and user */
    /* number from the user and fill in the rest of the parameters */
    /* with null values */
    printf("Enter the year: ");
    gets(user_query.year);

    printf("Enter the Eln: ");
    gets(user_query.eln);

    printf("Enter the seq no: ");
    gets(user_query.seq_no);

    printf("Enter the offset: ");
    gets(user_query.offset);

    printf("Enter the User No: ");
    gets(temp);
    user_no = atoi(temp);

    user_query.est[0] = 0;
    user_query.first[0] = 0;
    user_query.last[0] = 0;
    user_query.ssn[0] = 0;

    /* print the parameters on the screen for debugging purposes */

```

```

printf("%s\n%s\n%s\n", user_query.year, user_query.ein,
       user_query.seq_no, user_query.offset, user_no);

/* Open the user query file
   sprintf(qfilename, "query%d.txt", user_no);
   if(qfile == NULL)
   {
       printf("ERROR: Cannot open file %s\n", qfilename);
       exit(1);
   }

/* Write the query structure to the file for the server
   if(fwrite(&user_query, sizeof(struct USER_QUERY), 1, qfile) == 0)
   {
       printf("ERROR: Cannot write query\n");
       fclose(qfile);
       exit(1);
   }
   fclose(qfile);

/* Call the rpc function, and print the result
   res = callrpc(SERVER, 0x35000000, 1, 1, xdr_int, &user_no, xdr_int, &x);
   if(res)
   {
       cint_perrno(res);
       printf("%d\n", x);
       break;
   }
   case '6':
       printf("Blanket Info\n");
       /* Get the year, ein, seq # and user # from the user and
          /* fill in the rest of the parameters with null values
          printf("Enter the year: ");
          gets(user_query.year);
          printf("Enter the EIN: ");
          gets(user_query.ein);
          printf("Enter the seq no: ");
          gets(user_query.seq_no);
          printf("Enter the User No: ");
          gets(temp);
          user_no = atoi(temp);
          user_query.est[0] = 0;
          user_query.first[0] = 0;
          user_query.last[0] = 0;
          user_query.ssn[0] = 0;
          user_query.offset[0] = 0;

/* print the parameters on the screen for debugging purposes */
printf("%s\n%s\n", user_query.year, user_query.seq_no,
       user_no);

/* Open the user query file
   sprintf(qfilename, "query%d.txt", user_no);
   qfile = fopen(qfilename, "wb");
   if(qfile == NULL)
   {
       printf("ERROR: Cannot open file %s\n", qfilename);
       exit(1);
   }

/* Write the query structure to the file for the server
   if(fwrite(&user_query, sizeof(struct USER_QUERY), 1, qfile) == 0)
   {
       printf("ERROR: Cannot write query\n");
       fclose(qfile);
       exit(1);
   }
   fclose(qfile);

```



```

/* Call the rpc function, and print the result */
res = callrpc(SERVER,0x32000000,1,1,xdr_int,&user_no,xdr_int,&x);
if(res)
    cint_perrno(res);
else
    printf("%d\n", x);
break;
case '7':
    /* Get the year, ein, seq # and user # from the user and
    /* fill in the rest of the parameters with null values */
    printf("Print Report Selection\n");
    printf("Enter the year: ");
    gets(user_query.year);

    printf("Enter the EIN: ");
    gets(user_query.ein);

    printf("Enter the seq no: ");
    gets(user_query.seq_no);

    printf("Enter the User No: ");
    gets(temp);
    user_no = atoi(temp);

    user_query.est[0] = 0;
    user_query.seq_no[0] = 0;
    user_query.first[0] = 0;
    user_query.last[0] = 0;
    user_query.ssn[0] = 0;
    user_query.offset[0] = 0;

    /* print the parameters on the screen for debugging purposes */
    printf("%s\n%s\n%s\n", user_query.year, user_query.ein,
           user_query.offset, user_no);

    /* Open the user query file
    sprintf(qfilename, "query%d.txt", user_no);
    qfile = fopen(qfilename, "wb");
    if(qfile == NULL)
    {
        printf("ERROR: Cannot open file %s\n", qfilename);
        exit(1);
    }

    /* Write the query structure to the file for the server */
    if(fwrite(&user_query, sizeof(struct USER_QUERY), 1, qfile) == 0)
    {
        printf("ERROR: Cannot write query\n");
        fclose(qfile);
        exit(1);
    }
    fclose(qfile);

    /* Call the rpc function, and print the result */
    res = callrpc(SERVER,0x37000000,1,1,xdr_int,&user_no,xdr_int,&x);
    if(res)
        cint_perrno(res);
    else
        printf("%d\n", x);
    break;
}

/* parse_query()
*
* This function will parse out the first and last
* name from the name file string.
*
* Input: string containing namefile entry, "namestring"
*        user query structure, "user_query"
*/

```

```
*/
parse_query(namestring, user_query)
char namestring[];
struct USER_QUERY *user_query;
{
    int i, j;

    /* Skip past the actual name entry in the report */
    i = 0;
    while(namestring[i] != ' ')
        i++;

    /* pull out the last name
       i = i+3;
       j = i;
       while(namestring[j] != ' ')
           j++;

    /* pull out the first name
       while(namestring[j] == ' ')
           j++;

    /* copy the first and last names to the user query */
    strncpy(user_query->last, namestring+i, j-i);
    user_query->last[j-i] = 0;
    strncpy(user_query->first, namestring+j);
    user_query->ssn[0] = 0;
}
```

```

#
# Bottom Level Makefile (~/.ssapilot/src/bin/debug)
#
# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependencies for the source files.
#
# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../../..
LIBDIR       = $(PROJECT_ROOT)/lib
INCDIR       = $(PROJECT_ROOT)/include
BINDIR       = $(PROJECT_ROOT)/bin

# this is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE   = debug
SRC           = debug.c
LIBS          = $(LIBDIR)/libtest.a $(LIBDIR)/libgen_eamate.a
CLIBS         = -ltest -lgen_eamate -lm
OBJ           = debug.o
CFLAGS        = -I$(INCDIR) -L$(LIBDIR)
CC            = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependencies for each of the source files
depend : $(SRC)
    $(CC) -M $(CFLAGS) $(SRC) > dependlist
    sed -e '1,/^# DO NOT DELETE/id' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv mm.tmp Makefile.bak
    rm mm.tmp Makefile
    rm -f dependlist

```

```

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
    $(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(BINDIR)

.c.o :
    $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
debug.o: debug.c
debug.o: /usr/ucbinclude/stdio.h
debug.o: ../../../../include/params.h
debug.o: ../../../../include/eamatestruct.h
debug.o: ../../../../include/btreestruct.h

.c.o :
    $(CC) -c $(CFLAGS) $<

```



```

/*
 * debug.c
 * version 4.0
 * 10/29/93
 *
 * by Natalie Willman
 *
 */

/* Include Files */
#include <stdio.h>
#include "params.h"
#include "eamstruct.h"
#include "btreestruct.h"

/* Function Prototypes
char check_data();
char check_idx();
char display_unrec();
char make_search_table();
char rw_browse();
char rw_query();
char rw_browse_off();
char rw_btree();
char rw_detail();
char rw_dupfile();
char rw_emp_detail();
char rw_emplr();
char rw_emplr_idx();
char rw_error_search();
char rw_off();
char rw_parsectrl();
char rw_search_stats();
char translate_off();
*/

/* retrieve and open the output file name
printf("Enter the Output File Name: ");
gets(outfile);

out = fopen(outfile, "r");
if(out != NULL)
{
    printf("ERROR: This file already exists\n");
    fclose(out);
    exit(1);
}

out = fopen(outfile, "w");
if(out == NULL)
{
    printf("ERROR: Cannot open file %s\n", outfile);
    exit(1);
}

/* switch based upon the user's selection
switch(selection)
{
    case 'A':
        printf("Check Detail Files for All Data\n");
        printf("Enter the File Name containing the List of Detail Files: ");
        gets(filename);
        result = check_data(filename, out);
        if(result == ERROR)
            printf("ERROR: Could not complete operation\n");
        break;
    case 'B':
    case 'b':
        printf("Merge and Display Btree and Dupe File\n");
        printf("Enter the Btree File Name: ");
        gets(filename);
        printf("Enter the Dupe File Name: ");
        gets(filename);
        result = check_idx(filename, filename1, out);
        if(result == ERROR)
            printf("ERROR: Could not complete operation\n");
        break;
    case 'C':
    case 'c':
        printf("Display Unrecognized Character in Parse File\n");
        printf("Enter the COM File Name: ");
        gets(filename);
        printf("Enter the Offset: ");
        gets(offset);
        result = display_unrec(filename, offset, out);
*/

```

```

if (result == ERROR)
    printf("ERROR: Could not complete operation\n");
break;
case 'p':
case 'd':
    printf("Display Browse Data File\n");
    printf("Enter the Browse File Name: ");
    gets(filename);
    result = rw_browse(filename, out);
    if (result == ERROR)
        printf("ERROR: Could not complete operation\n");
break;
case 'e':
case 'e':
    printf("Display Browse Data File from an Offset\n");
    printf("Enter the Browse File Name: ");
    gets(filename);
    printf("Enter the Offset: ");
    result = rw_browse_off(filename, atol(offset), out);
    if (result == ERROR)
        printf("ERROR: Could not complete operation\n");
break;
case 'f':
case 'f':
    printf("Display a Btree File\n");
    printf("Enter the Btree File Name: ");
    gets(filename);
    result = rw_btree(filename, out);
    if (result == ERROR)
        printf("ERROR: Could not complete operation\n");
break;
case 'g':
case 'g':
    printf("Display a Detail File\n");
    printf("Enter the Detail File Name: ");
    gets(filename);
    result = rw_detail(filename, out);
    if (result == ERROR)
        printf("ERROR: Could not complete operation\n");
break;
case 'h':
case 'h':
    printf("Display a Duplicate Postings File\n");
    printf("Enter the Duplicate Posting File Name: ");
    gets(filename);
    result = rw_duplicate(filename, out);
    if (result == ERROR)
        printf("ERROR: Could not complete operation\n");
break;
case 'i':
case 'i':
    printf("Display an Employee Detail Record File\n");
    printf("Enter the Employee Detail File Name: ");
    gets(filename);
    result = rw_emp_detail(filename, out);
    if (result == ERROR)
        printf("ERROR: Could not complete operation\n");
break;
case 'j':
case 'j':
    printf("Display an Employer Header Info File\n");
    printf("Enter the Employer Header Info File Name: ");
    gets(filename);
    printf("Display a List of Browse Data from a List of Offsets\n");
    printf("Enter the Browse File Name: ");
    gets(filename);
    printf("Enter the Offset List File Name: ");

```

```

result = rw_emplr(filename, out);
if (result == ERROR)
    printf("ERROR: Could not complete operation\n");
break;
case 'k':
case 'k':
    printf("Display an Employer Index File\n");
    printf("Enter the Employer Index File Name: ");
    gets(filename);
    result = rw_emplr_idx(filename, out);
    if (result == ERROR)
        printf("ERROR: Could not complete operation\n");
break;
case 'l':
case 'l':
    printf("Check a Search Stats File for Errors\n");
    printf("Enter the Search Stats File Name: ");
    gets(filename);
    result = rw_error_search(filename, out);
    if (result == ERROR)
        printf("ERROR: Could not complete operation\n");
break;
case 'm':
case 'm':
    printf("Display an Offset List File\n");
    printf("Enter the Offset List File Name: ");
    gets(filename);
    result = rw_off(filename, out);
    if (result == ERROR)
        printf("ERROR: Could not complete operation\n");
break;
case 'n':
case 'n':
    printf("Display a Parse Control File\n");
    printf("Enter the Parse Control File Name: ");
    gets(filename);
    result = rw_parsectrl(filename, out);
    if (result == ERROR)
        printf("ERROR: Could not complete operation\n");
break;
case 'o':
case 'o':
    printf("Display a Search Statistics File\n");
    printf("Enter the Search Statistics File Name: ");
    gets(filename);
    result = rw_search_stats(filename, out);
    if (result == ERROR)
        printf("ERROR: Could not complete operation\n");
break;
case 'p':
case 'p':
    printf("Make a Search Statistics Table\n");
    printf("Enter the Search Statistics File Name: ");
    gets(filename);
    result = make_search_table(filename, out);
    if (result == ERROR)
        printf("ERROR: Could not complete operation\n");
break;
case 'q':
case 'q':
    printf("Display a List of Browse Data from a List of Offsets\n");
    printf("Enter the Browse File Name: ");
    gets(filename);
    printf("Enter the Offset List File Name: ");

```

```
gets(filename1);
result = translate_off(filename1, filename, out);
if(result == ERROR)
    printf("ERROR: Could not complete operation\n");
break;
case 'r':
case 'R':
    printf("Display a Query File\n");
    printf("Enter the Query File Name:  ");
    gets(filename);
    result = rw_query(filename, out);
    if(result == ERROR)
        printf("ERROR: Could not complete operation\n");
}

/* close the output file
fclose(out);
}
```



```

# Bottom Level Makefile (~/.ssapilot/src/bin/download)
#
# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependancies for the source files.
#
# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../../..
LIBDIR      = $(PROJECT_ROOT)/lib
INCDIR      = $(PROJECT_ROOT)/include
BINDIR      = $(PROJECT_ROOT)/bin

# this is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE  = download
SRC          = download.c
LIBS         =
CLIBS        =
OBJ          = download.o
CFLAGS       = -I$(INCDIR) -L$(LIBDIR)
CC           = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependancies for each of the source files
depend : $(SRC)
    $(CC) -M $(CFLAGS) $(SRC) > dependlist
    sed -e '1,/^# DO NOT DELETE/!d' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv mm.tmp Makefile.bak
    mv mm.tmp Makefile
    rm -f dependlist

```

```

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
    $(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(BINDIR)

.c.o :
    $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
download.o: download.c
download.o: /usr/include/stdio.h

```

```
/*
 * download.c
 * Version 1.0
 * 9/8/93
 *
 * Natalie Willman
 *
 * This program will operate the tape drive
 * to transfer a tape file to the magnetic
 * disk (using dd). A counter loop will allow
 * more than one file to be downloaded
 *
 */

#include <stdio.h>
#define TRUE 1
#define FALSE 0

main()
{
    int counter;          /* employer report sequence number */
    char done, command[100]; /* boolean flag, system command buffer */
    char response;        /* user response for done - y/n */

    done = FALSE;

    /* obtain the starting sequence number to use
    printf("Enter the counter value (to be used in filename)\n");
    scanf("%d%c", &counter);
    printf("counter = %d\n", counter);

    /* Prompt operator to load the tape and wait for ack from user */
    printf("Enter the tape, and press enter when ready ...\n");
    getchar();

    /* While there are still tapes to be downloaded, issue the dl
    /* command and the tape will download. Block info will be
    /* printed to the screen which will match the count on the
    /* tape label
    while (!done)
    {
        printf("current file = employer.%d\n", counter);
        sprintf(command, "dd if=/dev/rmt/0 of=employer.%d ibs=5187", counter);
        system(command);

        /* Increment the counter and prompt to see if user is done
        counter++;
        printf("Enter the next tape? (y/n) (If yes, make sure tape is loaded)\n");
        scanf("%c%c", &response);
        if (response != 'y')
            done = TRUE;
    }
}
```

```

#
# Bottom Level Makefile (-/ssapilot/src/bin/fix_parse)
#
# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependancies for the source files.
#
# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../../..
LIBDIR      = $(PROJECT_ROOT)/lib
INCDIR      = $(PROJECT_ROOT)/include
BINDIR      = $(PROJECT_ROOT)/bin

# this is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE = fix_parse
SRC         = fix_parse.c
LIBS        = $(LIBDIR)/libtest.a $(LIBDIR)/libgen_eamate.a
CLIBS       = -ltest -lgen_eamate -lm
OBJ         = fix_parse.o
CFLAGS      = -I$(INCDIR) -I$(LIBDIR)
CC          = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean : rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
      rm -f $(EXECUTABLE)
      rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependancies for each of the source files
depend : $(SRC)
      $(CC) -M $(CFLAGS) $(SRC) > dependlist
      sed -e '1,/^# DO NOT DELETE/!d' Makefile > mm.tmp
      cat dependlist >> mm.tmp
      mv Makefile Makefile.bak
      mv mm.tmp Makefile
      rm -f dependlist

```

```

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
      $(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
      cp $(EXECUTABLE) $(BINDIR)

.c.o : $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
debug.o: debug.c
debug.o: /usr/include/stdio.h
debug.o: ../../../../include/params.h
debug.o: ../../../../include/eamatestruct.h
debug.o: ../../../../include/btreestruct.h

```



```

/*
 * fix_parse.c
 * version 4.0
 * 1/29/94
 *
 * by Natalie Willman
 *
 *
 *
 */

/* Include Files */
#include <stdio.h>
#include "params.h"
#include "eamatestruct.h"
#include "btreesstruct.h"

/* Function Prototypes
char fix_parse_all();
char fix_parse_by_seq();
char change_browse();
char change_detail();
*/

/*
 * MAIN PROGRAM
 *
 * This program will present a list of options to the user. The
 * user will select which debug option is desired, and the proper
 * function will be called, and all output will be directed to a
 * user specified file
 */

main()
{
    char ein(20), seq[5];          /* Input filenames */
    char result,                  /* result from sub functs */
    selection;                    /* user selection */

    /* Print list of selections for the user */
    printf("A Remove an EIN/SEQ Combination\n");
    printf("NOTE: Only chose this selection if there will\n");
    printf("still exist an employer report for this ein\n");
    printf("B Remove an entire EIN\n");
    printf("C Change the Browse Location for an EIN\n");
    printf("Enter the Selection: ");
    scanf("%c%c", &selection);

    /* switch based upon the user's selection */
    switch(selection)
    {
        case 'A':
        case 'a':
            printf("Fix Parse for an EIN/SEQ Combination\n");
            printf("Enter the EIN: ");
            gets(ein);
            printf("Enter the Sequence Number: ");
            gets(seq);
            result = fix_parse_by_seq(ein, seq);
            if(result == ERROR)
                printf("ERROR: Could not complete operation\n");
            break;
        case 'B':
        case 'b':
            printf("Fix Parse for an entire EIN\n");

```

```

            printf("Enter the EIN: ");
            gets(ein);
            result = fix_parse_all(ein);
            if(result == ERROR)
                printf("ERROR: Could not complete operation\n");
            break;
        case 'C':
        case 'c':
            printf("Change the Browse Location for an EIN\n");
            printf("Enter the EIN: ");
            gets(ein);
            result = change_browse(ein);
            if(result == ERROR)
                printf("ERROR: Could not complete operation\n");
            break;
    }
}

```

```

#
# Bottom Level Makefile (~/.ssapilot/src/bin/index)
#
# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependancies for the source files.
#
# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../..
LIBDIR      = $(PROJECT_ROOT)/lib
INCDIR      = $(PROJECT_ROOT)/include
BINDIR      = $(PROJECT_ROOT)/bin

# this is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE  = index
SRC          = index.c
LIBS         = $(LIBDIR)/libgen_eamata.a $(LIBDIR)/libtree_data.a
CLIBS        = -lgen_eamata -lm -lbtrec_data
OBJ          = index.o
CFLAGS       = -IS $(INCDIR) -LS $(LIBDIR)
CC           = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependancies for each of the source files
depend : $(SRC)
    $(CC) -M $(CFLAGS) $(SRC) > dependlist
    sed -e '1,/^# DO NOT DELETE/id' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv Makefile Makefile.bak
    mv mm.tmp Makefile
    rm -f dependlist

```

```

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
    $(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(BINDIR)

.c.o :
    $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
index.o : index.c
index.o : /usr/include/stdio.h
index.o : /usr/include/math.h
index.o : /usr/include/float.h
index.o : /usr/include/sys/ieee.h
index.o : /usr/include/sys/types.h
index.o : /usr/include/sys/select.h
index.o : /usr/include/sys/time.h
index.o : /usr/include/sys/types.h
index.o : /usr/include/time.h
index.o : /usr/include/sys/symmacros.h
index.o : /usr/include/sys/time.h
index.o : ../include/params.h
index.o : ../include/eamestruct.h
index.o : ../include/btreestruct.h

```

```

/*
 * Index.c
 * version 4.0
 * 11/03/93
 *
 * by Natalie Willman
 *
 * This module contains the main indexing control routine, and its
 * auxiliary functions.
 *
 * THE LIST OF FILES TO INDEX MUST BE JUST THE BROWSE FILENAME, NOT
 * INCLUDING A PATH NAME. BROWSE FILES ARE ASSUMED TO HAVE A LOGICAL
 * LINK IN THE ./1991BRW DIRECTORY. THE PROGRAM MUST BE RUN FROM THE
 * ROOT DIRECTORY OF THE SSA ACCOUNT
 */

/* Include Files */
#include <stdio.h>
#include <math.h>
#include <sys/types.h>
#include <sys/time.h>
#include "params.h"
#include "ematestruct.h" /* Data file structure definitions */
#include "btreetstruct.h" /* Btree file structure definitions */

/*
 * MAIN PROGRAM
 *
 * This function takes as input a list of data files of employee browse
 * records generated by the parse.c program.
 *
 * Each file is passed to index_data(), where it will be indexed.
 */

main()
{
    char filename[FILENAME]; /* Name of browse file to open */
    char year[5]; /* Year of the data being indexed */
    FILE *filelist; /* Pointer to file of filenames */
    FILE *statslog; /* Pointer to index statistics file */

    /* Open the file of stats on index process.
     * If the file does not exist, then open it to write, and
     * append a descriptor header. If it did exist, then close
     * the file, and open it to append at the end of file
     */
    statslog = fopen("indexstats", "r+");
    if (statslog == NULL)
    {
        statslog = fopen("indexstats", "w+");
        if (statslog == NULL)
            error_exit("Cannot open stats log for index");
        fprintf(statslog, "%12s%10s%10s%10s%10s%10s\n",
            "EIN", "NUM RECS", "CTIME", "ETIME", "TMP DSK",
            "NAME IDX", "NAME DUP", "SSN IDX", "SSN DUP");
    }
    else
    {
        fclose(statslog);
        statslog = fopen("indexstats", "a+");
    }

    /* Get the year of the data to be indexed
     * printf("Enter the year of the data: ");
     * scanf("%s%c", year);
     */

    /* Open the file of names of text files to index
     * filelist = fopen("einlist.index", "r");
     * if (filelist == NULL)
     *     error_exit("No list of ein's to index");
     */

    /* Index the data ...
     * while (fscanf(filelist, "%s", filename) != EOF)
     *     index_data(filename, year, statslog);
     */

    /* close the file
     * fclose(statslog);
     * fclose(filelist);
     */

    /*
     * index_data()
     *
     * This function is the main control function to index a data file.
     * The file will be read, and an index file for last name field and ssn
     * field (both broken into n-grams) will be generated.
     * If specified by the DUPEFILE parameter, a duplicate gram file will
     * be generated, otherwise all duplicate grams will be put in the btree.
     *
     * Input: the filename of the data file, filename.
     * the year of the data, year.
     * file pointer to the stats file, statslog.
     *
     * output: The data files are indexed, creating .name.idx, .name.dup,
     * .ssn.idx, .ssn.dup
     */

    index_data(filename, year, statslog)
    {
        char fname[];
        char year[];
        FILE *statslog;
        {
            FILE *record_fil, /* File pointer to data file of records */
            *name_tree, /* File pointer to the name btree */
            *ssn_tree, /* File pointer to the ssn btree */
            *name_dupe, /* File pointer to the name duplicate file */
            *ssn_dupe, /* File pointer to the ssn duplicate file */
            *statsfile; /* File pointer to the gram statistics file */
            struct EMATE W2EMPL BRW record, *precord; /* brw data - employee */
            struct MEMNODE *nameroot, /* pointer to root node in name btree */
            *ssnroot; /* pointer to root node in ssn btree */
            long browse_loc; /* file offset in brw file of empl info */
            char parse_array[ARRAY_SIZE[KEYLEN]]; /* array to hold ngrams */
            short count, i; /* counter variables */
            long name_loc, ssn_loc; /* file offset in name/ssn btrees */
            char trash[COMMANDLENGTH]; /* trash character string (for concat) */
            struct STATS stat, *pstat; /* gram statistics gathering structure */
            short numgrams; /* numgrams from parse, numwords in rec */
            long recount; /* total word count in file */
            long timerst, timerend; /* timer vars for "stopwatch" */
            char ein[20]; /* ein associated with the data file */
            char filename[50]; /* complete browse filename, including path */
            time_t etimerst, etimerend, junk; /* elapsed timer variables */

```



```

long tmpdisk, nameidx, /* size of files and memory */
namedup, ssnidx, ssnidup;
double tmpkilo;

/* Create the real browse filename by adding the path to the name */
sprintf(filename, "1991BRW/%s", name);

/* Open the data file to be indexed
record_fil = fopen(filename, "r");
if(record_fil == NULL)
error_exit("Error opening data file");

/* Initialize the employee browse record structure
precord = &record;

/* Initialize the gram stats counter, if gram stats are being kept */
if(KEEPSTATS == TRUE)
{
pstat = &stat;
for(i = 0; i < MAXGRAMS; i++)
stat.numgrams[i] = 0;
}

/* start stopwatch for index timing (cpu and elapsed time)
timestr = clock();
etimerst = time(&junk);

/* create the ein from the browse filename
create_ein(ein, name);

/* make the base filename for indexing files
create_filename(filename, year, ein);

/* create the b+ trees for the name and SSN btree files -- filenames */
/* are XXXX.YY-YYYYYY.name.idx and XXXX.YY-YYYYYY.ssn.idx (see */
/* above). Initialize the "write location" variables for each index.*/
concat(trash, filename, ".name.idx");
name_tree = fopen(trash, "w+");
name_loc = 0;
nameroot = btree_create(&name_loc);

concat(trash, filename, ".ssn.idx");
ssn_tree = fopen(trash, "w+");
ssn_loc = 0;
ssnroot = btree_create(&ssn_loc);

/* If a duplicate listing file is requested, then create the files */
/* for the name and ssn. File names are XXXX.YY-YYYYYY.name.tmp */
/* and XXXX.YY-YYYYYY.ssn.tmp. These will later be written to */
/* files with extension .dup
if(DUPEFILE == TRUE)
{
concat(trash, filename, ".name.tmp");
name_dupe = fopen(trash, "w+");
concat(trash, filename, ".ssn.tmp");
ssn_dupe = fopen(trash, "w+");
}

/* set record count value to zero, & initialize the browse_loc
reccount = 0;
browse_loc = 0;

/* While there are records in the record file ...
while(browse_loc != -1)
{
/* Read a record into "precord", and store its location in the */
/* data file. If the location returned is not a -1 (indicating */
/* that EOF has been reached), then parse the name and ssn fields */
/* into ngrams, and insert the ngrams into the B+ tree
browse_loc = read_rec(precord, record_fil);
if(browse_loc != -1)
{
reccount++;
numgrams = parse_name(precord->name, parse_array);
if(KEEPSTATS == TRUE) /* update gram stats counts for rec */
update_stats(numgrams, pstat, parse_array);
for(i = 0; i < numgrams; i++)
btree_insert(&nameroot, parse_array[i], browse_loc, name_dupe,
&name_loc, numgrams);
count = parse_ssn(precord->ssn, parse_array);
for(i = 0; i < count; i++)
btree_insert(&ssnroot, parse_array[i], browse_loc, ssn_dupe,
&ssn_loc, count);
}
}

/* update the statistics file
if(KEEPSTATS == TRUE)
{
concat(trash, filename, ".dat");
statsfile = fopen(trash, "w");
if(statsfile == NULL)
error_exit("Error opening the statistics file");
if(fwrite(pstat, sizeof(struct STATS), 1, statsfile) == 0)
error_exit("Error writing current ngram stats");
fclose(statsfile);
}

/* end timers
timerend = clock();
etimerend = time(&junk);

/* If there is a duplicate file, restructure the duplicate file */
/* from a linked list into a count followed by count offsets to */
/* records. Close and remove the old duplicate file.
if(DUPEFILE == TRUE)
{
concat(trash, filename, ".name");
restruct_dupfile(trash, nameroot, name_dupe, &namedup);
concat(trash, filename, ".ssn");
restruct_dupfile(trash, ssnroot, ssn_dupe, &ssnidup);
fseek(name_dupe, 0L, 2);
fseek(ssn_dupe, 0L, 2);
tmpdisk = ftell(name_dupe) + ftell(ssn_dupe);
fclose(name_dupe);
fclose(ssn_dupe);
}

/* walk the btree and write the nodes to a file
btree_walk(nameroot, name_tree);
btree_walk(ssnroot, ssn_tree);

/* Close the B+ tree file and the data file
fseek(name_tree, 0L, 2);
fseek(ssn_tree, 0L, 2);
nameidx = ftell(name_tree);

```

```

ssnidx = ftell(ssn_tree);
fclose(ssn_tree);
fclose(ssn_tree);
fclose(record_fil);

/* If a duplicate posting file was being created, remove temp files */
if (DUPEFILE == TRUE)
{
    printf(trash, "%s%s", "rm ", filename, ".name.tmp");
    system(trash);
    printf(trash, "%s%s", "rm ", filename, ".ssn.tmp");
    system(trash);
}

if (tmpdisk != 0)
    tmpkilo = (double) tmpdisk / (double) 1024;
else
    tmpkilo = 0.000;

/* Print the index stats to the statistics file */
printf(statslog, "%12s%10d%10.2f%10.2f%10.2f%10.2f\n",
    filename, 5, reccount, (double) (timerend-timerst) / (double) 1000000,
    etimerend-etimerst, tmpkilo, (double) nameldx / (double) 1024,
    (double) namedup / (double) 1024, (double) ssnidx / (double) 1024,
    (double) ssndup / (double) 1024);
}

/*
 * update stats
 *
 * this function updates the gram stats with a count of the
 * number of each gram encountered in this record
 *
 * Input: The number of grams in the parse array
 *         The pointer to the gram statistics structure
 *         The array containing the ngrams
 */
update_stats(numgrams, pstat, parse_array)
struct STATS *pstat;
char parse_array[ARRAY_SIZE][MAX_GRAM_SIZE+1];
short numgrams;
{
    int j;
    int gramnum;

    pstat->count++; /* increment number of records viewed */
    for (j = 0; j < numgrams; j++)
    {
        gramnum = gram_to_int(parse_array[j]); /* convert gram to int and */
        pstat->numgrams[gramnum]++; /* increment counter for it */
    }
}

/*
 * reconstruct_dupfile
 *
 * this function takes a duplicate file and restructures it from
 * a linked list format to a count of each type of ngram, followed
 */
by a list of count offsets
 *
 * Input: Base duplicate file name
 *         Pointer to the root of the tree
 *         Pointer to the unorganized dupe file
 *         The size of the duplicate file
 */
restruct_dupfile(filename, root, dupe_file, length)
char *filename;
FILE *dupe_file;
struct MEMNODE *root;
long *length;
{
    FILE *order_dupe; /* pointer to new duplicate file */
    long dupoffset, countloc; /* file offsets */
    int i, j; /* counter variables */
    char trash[COMMANDLENGTH]; /* trash char array */
    struct DUPELIST dupelist; /* structure to hold dupes */
    long count; /* count of dupes for a gram */

    /* create and open the new ordered duplicate file */
    concat(trash, filename, ".dup");
    order_dupe = fopen(trash, "wt");

    /* read the tree until first leaf is found (start of leaf list) */
    while (root->leaf != TRUE)
        root = root->branch.mem[0];

    /* for each key in the tree, read the duplicate list, and write */
    /* to ordered file */
    do
    {
        for (i = 0; i < root->count; i++)
        {
            /* mark location of the count for dupes,
            countloc = ftell(order_dupe);
            count = root->num_dupe[i];

            /* copy the memory linked list for the gram into an array */
            /* structure, and write the array to the disk */
            for (j = 0; j < root->num_dupe[i]; j++)
            {
                dupelist.record[j].dupe_offset = (root->branch.dup[i])->offset;
                dupelist.record[j].dupe_weight = (root->branch.dup[i])->weight;
                root->branch.dup[i] = {root->branch.dup[i]}->next_key;
            }
            if (fwrite(dupelist.record, sizeof(struct RECINFO),
                (int) root->num_dupe[i], order_dupe) == 0)
                error_exit("error reorganizing dupe file, 2");
        }

        /* get the offset to the next list for this gram in the temp */
        /* duplicate file */
        dupoffset = root->dupe_offset[i];

        /* read in each array of offsets for the gram, and write it */
        /* to the new dupe file. Increment the count variable. */
        while (dupoffset != -1)
        {
            fseek(dupe_file, (long) dupoffset, 0);
            if (fread(&dupelist, sizeof(struct DUPELIST), 1, dupe_file) == 0)
                error_exit("error reading old dupe file");
        }
    }
}

```

```

    if( fwrite(dupelist.record, sizeof(struct RECINFO), MAXDUPE,
        order_dupe) == 0)
        error_exit("error writing to new dupe file");

    dupoffset = dupelist.next_set;
    count = count + MAXDUPE;
}

/* mark next area in the file to which to write, and go to */
/* count location and write out the new count (if there were */
/* more offsets than just those in memory). Then go back */
/* to the EOF location. Write the new duplicate list offset */
/* to the node branch
if(count != root->num_dupe[i])
    root->num_dupe[i] = count;
root->branch.disk[i] = countloc;
}

/* read in the next node if it exists
root = root->branch.mem[MAXCHILD];
while(root != NULL);

/* close the new duplicate file
*length = ftell(order_dupe);
fclose(order_dupe);
}

/*
* read_rec()
*
* This function reads a record to be indexed from a data file and
* stores it in a structure of type REC. It returns a long value
* which represents the file offset location of the record in the
* data file. All records except employee information records are
* ignored.
*
* Input: Pointer to structure to hold data record, "plndex_rec"
*        Pointer to data file containing records, "record_fil"
* Output: File offset of record in browse file
*
*/
read_rec(plndex_rec, record_fil)
    struct _EAMATE_W2EMPL_BRW *plndex_rec;
    FILE *record_fil;
{
    long loc;
    int rcread;

    /* While the record file is not empty, and the search for the */
    /* next employee information record is not done ... */
    /* record the current location in the file */
    loc = ftell(record_fil);
    rcread = read_eamate_w2employee_browse(plndex_rec, record_fil);

    if(rcread == TRUE)
        return(loc);
    else
        return(-1);
}

```



```

#
# Bottom Level Makefile (~/.ssapilot/src/bin/indexemplr)
#
# This make file is at the lowest level in the
# project hierarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependencies for the source files.
#
# This is a list of the key directories in the
# project hierarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../../..
LIBDIR      = $(PROJECT_ROOT)/lib
INCDIR      = $(PROJECT_ROOT)/include
BINDIR      = $(PROJECT_ROOT)/bin

# this is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE  = indexemplr
SRC          = indexemplr.c
LIBS         = $(LIBDIR)/libgen eamdate.a $(LIBDIR)/libtree_emplr.a
CLIBS        = -lgen_eamdate -lm -lbtree_emplr
OBJ          = indexemplr.o
CFLAGS       = -I$(INCDIR) -L$(LIBDIR)
CC           = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependencies for each of the source files
depend : $(SRC)
    $(CC) -M $(CFLAGS) $(SRC) > dependlist
    sed -e '1,/^# DO NOT DELETE/!d' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv Makefile Makefile.bak
    mv mm.tmp Makefile
    rm -f dependlist

```

```

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
    $(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(BINDIR)

.c.o :
    $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
indexemplr.o: indexemplr.c
indexemplr.o: /usr/include/stdio.h
indexemplr.o: ../../include/paramsempir.h
indexemplr.o: ../../include/eamteststruct.h
indexemplr.o: ../../include/btreestruct.h

```

```

/*
 * Indexemplr.c
 * version 3.0
 * 09/08/93
 *
 * by Natalie Willman
 *
 * This module contains the main indexing control routine, and its
 * auxiliary functions to index the employer information. It must
 * be linked with btree_emplr library
 */

/* Include Files */
#include <stdio.h>
#include "paramsempir.h" /* employer data structure and parameters */
#include "eamatstruct.h" /* employer data structure and parameters */
#include "btreestruct.h" /* btree data structures and parameters */

/*
 * MAIN PROGRAM
 *
 * This function takes as input a data file of employer information
 * records created when data records are indexed.
 *
 * The file will be read, and an index file (indexed on the ein field),
 * will be generated.
 */

main()
{
    FILE *record_file, /* File pointer to data file of records */
        *tree; /* File pointer to the btree file */
    struct EAMATE_W2EMPLR_INFO record, *precord; /* employer headers */
    struct MENNODE *root; /* pointer to root node in b+ tree */
    long loc; /* file offset in file of employer info */
    long tree_loc; /* file offset in btree file */
    long read_rec();

    /* open the employers file */
    record_file = fopen("1991.employers.text", "r");
    if(record_file == NULL)
    {
        printf("ERROR: Cannot open file 1991.employers.text\n");
        exit(1);
    }

    /* Initialize the Record structure and the employer structure */
    precord = &record;

    /* create the b+ tree */
    tree = fopen("1991.employers.idx", "w+");
    tree_loc = 0;
    root = btree_create(&tree_loc);

    /* While there are records in the record file ...
       while(!feof(record_file))
    {
        /* Read a record into "precord", and store its location in the
         * data file. If the location returned is not a -1 (indicating
         * that EOF has been reached), then insert the record into the
         * B+ tree
    }
    */
    while (1)
    {
        /* read_rec()
         * This function reads a record to be indexed from a data file of
         * employer information. It returns a long value which represents
         * the file offset location of the record in the data file.
         *
         * Input: Pointer to structure to hold data record, "pindex_rec"
         * Pointer to data file containing records, "record_file"
         * Output: File offset of record in browse file
        */
        long rec_loc = read_rec(precord, record_file);
        if(rec_loc != -1)
        {
            btree_insert(&root, precord->ein, loc, NULL, &tree_loc, 1);
        }
    }

    /* Walk the btree and sort the employers to an output file
       inorder_walk(root, tree);

    /* close the btree file and the data file
       fclose(tree);
       fclose(record_file);
    */

    /* inorder_walk()
     *
     * This function walks the btree and prints each of the data in the
     * nodes to a file in sorted order
     *
     * Input: The root of the tree in memory, "root"
     * The file pointer to the final file, "tree"
     *
     * Output: The Tree data is written to a file
    */

    inorder_walk(root, tree);
    struct MENNODE *root;
    FILE *tree;
    {
        int i;
        struct EMPPLR_IDX rec;

        /* go to the first entry in the nodes
           while(root->leaf != TRUE)
           {
               root = root->branch.mem[0];
           }

        /* walk thru each of the nodes and print the data in each node
           do
           {
               for(i = 0; i < root->count; i++)
               {
                   strcpy(rec.ein, root->key[i]);
                   rec.offset = root->branch.disk[i];
                   fwrite(&rec, sizeof(struct EMPPLR_IDX), 1, tree);
               }
               root = root->branch.mem[MAXCHILD];
           }
           while(root != NULL);
        */
        read_rec()
    }
}

```

```
*/
long read_rec(pindex_rec, record_fll)
struct EMATE_W2EMPLR_INFO *pindex_rec;
FILE *record_fll;
{
    long loc;
    int num;

    /* record the current location in the file */
    loc = ftell(record_fll);

    /* read in an employer header record */
    num = fread(pindex_rec, sizeof(struct EMATE_W2EMPLR_INFO), 1, record_fll);

    /* if one exists, return the location in the file, otherwise */
    /* return -1 */
    if (num != 0)
        return(loc);
    else
        return(-1);
}

/* error exit ()
*
* This function accepts as input an error message, and prints the
* error message and exits
*
* Input: error message character string
* Output: prints the error message, and exits
*
*/
void error_exit(message)
char message[];
{
    printf("%s\n", message);
    exit(1);
}
```



```

#
# Bottom Level Makefile (~/.ssapilot/src/bin/parse)
#
# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependencies for the source files.
#
# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../..
LIBDIR      = $(PROJECT_ROOT)/lib
INCDIR      = $(PROJECT_ROOT)/include
BINDIR      = $(PROJECT_ROOT)/bin

# This is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE  = parse
SRC          = parse.c
LIBS        = $(LIBDIR)/libgen_eamate.a
CLIBS       = -lgen_eamate -lm
OBJ         = parse.o
CFLAGS      = -I$(INCDIR) -L$(LIBDIR)
CC          = cc

# this make directive actually compiles the
# source files to executables
lc : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependencies for each of the source files
depend : $(SRC)
    $(CC) -M $(CFLAGS) $(SRC) > dependlist
    sed -e '1,/^# DO NOT DELETE/id' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv Makefile Makefile.bak
    mv mm.tmp Makefile
    rm -f dependlist

```

```

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
    $(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(BINDIR)

.c.o :
    $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
parse.o: parse.c
    parse.o: /usr/ucblib/include/stdio.h
    parse.o: ../../include/params.h
    parse.o: ../../include/eamatestruct.h

```

```

/* Define statements
#define RECLEN 134      /* 1+ true record length of data */

main()
{
    char filename[FILENAME], /* path and name of the COM file */
        year[5];           /* year of the data (1991) */
    FILE *parsectrl,        /* ptr to the parse control file */
        *filelist,         /* ptr to the list of COM files */
        *statsfile,        /* ptr to the report stats file */
        *debug;            /* ptr to the debug file */
    long platter_side;      /* platter/side for detail file */
    char browse_loc[3];     /* directory for browse file */

    /* Open the parse control file to read and write. If it is
    /* not yet created (parsectrl == NULL), then open it to write
    /* and read
    parsectrl = fopen("parsectrl", "r+");
    if(parsectrl == NULL)
    {
        parsectrl = fopen("parsectrl", "w+");
        if(parsectrl == NULL)
            error_exit("Error creating the parse control file");
    }

    /* Obtain from the operator the year of the data
    printf("Enter the year of the data: ");
    scanf("%s%c", year);

    /* Open the file of names of COM files to parse, including the
    /* platter/side for detail data and the browse location
    filelist = fopen("einlist.parse", "r");
    if(filelist == NULL)
        error_exit("No list of COM files to parse");

    /* Open the file of stats on employer reports to read & write.
    /* If the file does not exist, then open it to write, and
    /* append a descriptor header. If it did exist, then close
    /* the file, and open it to append at the end of file
    statsfile = fopen("einstats", "r+");
    if(statsfile == NULL)
    {
        statsfile = fopen("einstats", "w+");
        if(statsfile == NULL)
            error_exit("cannot create the stats file");
        fprintf(statsfile, "%12s%10s%10s%10s%10s%11s%13s%9s%9s%9s\n",
            "EIN", "SEQ", "NUM RECS", "WORD CT",
            "AVG WC", "TXT SIZE", "COM SIZE", "TXT/COM",
            "BROWSE SIZE", "BR/COM", "BR_LOC", "DET_LOC");
    }
    else
    {
        fclose(statsfile);
        statsfile = fopen("einstats", "a+");
    }

    /* open/create the debug file for monitoring the parse process
    debug = fopen("parse.debug", "a");
    if(debug == NULL)
        error_exit("error opening parse.debug file");

    /* Parse the data ...
    while(fscanf(filelist, "%s%d%s%c", filename, &platter_side,
        browse_loc) != EOF)

```

```

    parse_it(filename, year, platter_side, browse_loc, parsectrl,
        statsfile, debug);

/* Close the files that we opened
fclose(parsectrl);
fclose(debug);
fclose(statsfile);
fclose(filelist);
}

/*
*/

/* This function will take a COM file and will generate a text file
* from it, as well as creating a browse data file, and updating several
* statistics files.
*
* Input:  the name of the COM file, "filename"
*         the year of the data, "year"
*         the platter/side to store the detail file, "platter_side"
*         the subdir for the browse file, "browse_loc"
*         the ptr to the parsectrl file, "parsectrl"
*         the ptr to the stats file, "statsfile"
*         the ptr to the debug file, "debug"
*
* Output: the detail file and the browse file are created, the parse
*         control file, the stats file and the debug file are updated,
*         and the employer information file is updated.
*/

parse_it(filename, year, platter_side, browse_loc, parsectrl, statsfile, debug)
char filename[]; /* filename of COM file */
char year[]; /* the year of the data */
long platter_side; /* the platter/side for detail data */
char browse_loc[]; /* the subdir for the browse data */
FILE *parsectrl; /* file ptr to parse ctrl file */
/* debug, /* file ptr to debug file */
/* statsfile; /* file ptr to stats file */
{
    char ssarec[RECLEN]; /* array to hold COM file rec */
    datafile[FILENAME], /* filename of browse data file */
    namefile[FILENAME], /* filename of list of names */
    ssnfile[FILENAME], /* filename of list of ssns */
    detailfile[FILENAME], /* filename of the detail file */
    emplrfile[FILENAME], /* filename of employer info */
    curr_mrn[12]; /* current mrn of report */
    FILE *com, *data, *names, *brw; /* file ptrs for above */
    FILE *emplr_fil, *ssns; /* files and browse file */
    long recloc; /* record offset - for error records */
    /* data structures for the detail file and the browse file */
    struct EMATE_W2EMPLR_INFO emplr_info, *pemplr_info;
    struct EMATE_W2EMPLR_HEADER emplr_rec, *pemplr_rec;
    struct EMATE_W2EMPLR_DETAIL empl_rec, *pempl_rec;
    struct EMATE_W2EMPLR_BRW empl_brw, *pempl_brw;
    struct EMATE_W2INTERMED_TOT inter_rec, *pinter_rec;
    struct EMATE_W2FINAL_TOT final_rec, *pfinal_rec;
    struct EMATE_W2CUMEN_TOT cum_rec, *pcum_ein;
    struct CTRL_FILE ctrl_rec; /* Parse Control File struct */
    long recount, wc; /* count of records and words in name */
    long found, newreport; /* counter vars */

    if(ssarec[0] != '1')
    {
        printf("ERROR: File inconsistency in %s -- 1st record not an employer header\n",
            filename);
        fprintf(debug, "ERROR: File inconsistency in %s -- 1st record not an employer header\n",
            filename);
        fclose(com);
        fclose(emplr_fil);
        return(1);
    }
    else
    {
        /* Pull out the EIN, and use it as filename for data file */
        strncpy(datafile, ssarec+12, 10);
        datafile[10] = 0;

        /* Initialize the counter variables ...
        startcomsize = 0; /* start point for the com file size */
        recount = 0; /* total record count for the report */
        wc = 0; /* total report word count(name field) */
        found = FALSE;
        newreport = FALSE;
    }
}

```





```

    {
        strncpy(datafile, ssarec12, 10);
        datafile[10] = 0;
        newreport = FALSE;
        recount = 0;
        wc = 0;
        found = FALSE;
        fseek(parsectrl, 0L, 0);
        while( (!feof(parsectrl)) && (found == FALSE) )
        {
            ctrl_loc = ftell(parsectrl);
            fread((char *)ctrl_rec, sizeof(struct CTRL_FILE), 1, parsectrl);
            if(!feof(parsectrl))
            {
                if(strncmp(ctrl_rec.ein, datafile) == 0)
                {
                    found = TRUE;
                    interval = seq_to_int(ctrl_rec.seq);
                    interval++;
                    int_to_seq(interval, pemplr_rec->seq_no);
                    strcpy(ctrl_rec.seq, pemplr_rec->seq_no);
                    strcpy(browse_loc, ctrl_rec.browse_loc);
                    printf("Previous report for employer %s already parsed -- browse locati
on is %s\n",
                        ctrl_rec.ein, ctrl_rec.browse_loc);
                    fprintf(debug, "Previous report for employer %s already parsed -- brows
e location is %s\n",
                        ctrl_rec.ein, ctrl_rec.browse_loc);
                    fseek(parsectrl, ctrl_loc, 0);
                    fwrite((char *)ctrl_rec, sizeof(struct CTRL_FILE), 1, parsectrl);
                }
            }
            if(found == FALSE)
            {
                strcpy(ctrl_rec.ein, datafile);
                strcpy(ctrl_rec.seq, "AAA");
                strcpy(ctrl_rec.browse_loc, browse_loc);
                fseek(parsectrl, 0L, 2);
                fwrite((char *)ctrl_rec, sizeof(struct CTRL_FILE), 1, parsectrl);
            }
        }
        printf("Parsing file %s as EIN: %s SEQ: %s\n", filename, ctrl_rec.ein, ctr
l_rec.seq);
        fprintf(debug, "Parsing file %s as EIN: %s SEQ: %s\n", filename, ctrl_rec.
ein, ctrl_rec.seq);
        strcpy(namefile, datafile);
        strcpy(ssnfile, datafile);
        cr_browse_filename(datafile, year, browse_loc, namefile);
        cr_detail_filename(detailfile, year, namefile, ctrl_rec.seq, platter_side);
        data = fopen(detailfile, "w");
        if(data == NULL)
        {
            printf("ERROR: Cannot open file %s\n", detailfile);
            fprintf(debug, "ERROR: Cannot open file %s\n", detailfile);
            fclose(com);
            return(1);
        }
        brw = fopen(datafile, "a");
        if(brw == NULL)
        {
            printf("ERROR: Cannot open file %s\n", datafile);
        }
    }
}

fprintf(debug, "ERROR: Cannot open file %s\n", datafile);
fclose(com);
fclose(emplr_fll);
return(1);
}

pemplr_info->browse_start = 1 + (ftell(brw)/sizeof(struct EAMATE_W2EMPL_BRW));

strcat(namefile, ".names");
names = fopen(namefile, "a");
if(names == NULL)
{
    printf("ERROR: Cannot open file %s\n", namefile);
    fprintf(debug, "ERROR: Cannot open file %s\n", namefile);
    fclose(com);
    fclose(emplr_fll);
    fclose(data);
    fclose(brw);
    return(1);
}

strcat(ssnfile, ".ssns");
ssns = fopen(ssnfile, "a");
if(ssns == NULL)
{
    printf("ERROR: Cannot open file %s\n", ssnfile);
    fprintf(debug, "ERROR: Cannot open file %s\n", ssnfile);
    fclose(com);
    fclose(emplr_fll);
    fclose(data);
    fclose(brw);
    fclose(names);
    return(1);
}

/* Parse the employer record, and write out as 1st record */
/* In the data file
parse employer rec(ssarec, com, curr_mrn, pemplr_rec);
write_eamate_w2header(data, pemplr_rec);
}
else
{
    trash_employer_header(com);
}
else if(ssarec[42] == 'P') /* Employee record labels
trash_record_labels(com);
else if(ssarec[2] == 'O') /* Intermediate Total record
{
    parse_interm_rec(ssarec, com, pinter_rec);
    write_eamate_w2inter_total(data, pinter_rec);
}
else if(ssarec[4] == 'O') /* Final Total Record
{
    pemplr_rec->final_offset = ftell(data);
    parse_total_rec(ssarec, com, pfinal_rec);
    write_eamate_w2final_total(data, pfinal_rec);
    endcomsize = ftell(com);
    update_emplr_header(data, curr_mrn, pemplr_rec, recount, platter_side, ctrl_rec
.seq);
    cp_eamate_w2header(pemplr_rec, pemplr_info);
    cp_eamate_w2final(final_rec, pemplr_info);
}
}

```

```

    pempr_info->initials = INITIALSCALE * (wc/(double) pempr_rec->num_recs);
    write_emate_w2header_info(empr_fil, pempr_info);
    fclose(brw);
    fclose(names);
    fclose(ssns);
    fseek(data, 0L, 2);
    textsize = (double) ftell(data)/(double)1024;
    comsize = (double) (endcomsize-startcomsize)/(double) 1024;
    browsize = (double) (pempr_rec->num_recs*100)/(double)1024;
    fprintf(statsfile, "%12s%5s%10d%9.3f%10.2f%11.3f%13.2f%9.3f%9s%9d\n",
        pempr_rec->ein, pempr_rec->seq_no, pempr_rec->num_recs,
        wc, (double) wc/(double) pempr_rec->num_recs,
        textsize, comsize, textsize/comsize, browsize,
        browsize/comsize, browse_loc, platter_side);
    fclose(data);
    newreport = TRUE;
    startcomsize = endcomsize;
}
else if(ssarec[3] == 'C') /* Cumulative EIN total record */
{
    pempr_rec->cum_offset = ftell(data);
    parse_cum_ein(ssarec, com, pcum_ein);
    /* write_emate_w2cum_ein_total(data, pcum_ein); */
}
else if(ssarec[0] == 'A'); /* Trash records */
else if(ssarec[0] == 'E');
else if(ssarec[0] == 'F');
else
{
    printf("ERROR: Unidentified Record in %s at loc %d\n%s\n",
        filename, recloc, ssarec);
    fprintf(debug, "ERROR: Unidentified Record in %s at loc %d\n%s\n",
        filename, recloc, ssarec);
    return;
}
}

/* close browse, data and name files
fclose(data);
fclose(brw);
fclose(names);
fclose(ssns);
}

/* close COM file
fclose(com);
fclose(empr_fil);
}

/* parse_employee_record
*
* This function reads in the employee record, and parses the
* data from the trash, and fills the employee data structure
*
* Input: Array containing first line of the record, ssarec.
* Name file pointer, names.
* SSN file pointer, ssns.
* COM file pointer, empr.
* string containing the current MRN, curr_mrn.
* employee data structure, pempr_rec.
*/

```

```

*
* Output: Data structure record is filled and printed to the
* data file.
*
*/

parse_employee_rec(ssarec, names, ssns, emplr, curr_mrn, pempr_rec)
char ssarec[], curr_mrn[];
FILE *emplr, *names, *ssns;
struct EMATE_W2EMPL_DETAIL *pempr_rec;
{
    strcpy(pempr_rec->mrn, curr_mrn);
    /* Parse the first line of the employee record */
    strcpy(pempr_rec->ssn, ssarec+2, 11);
    pempr_rec->ssn[11] = 0;
    printf(ssns, "%s\n", pempr_rec->ssn);
    strcpy(pempr_rec->name, ssarec+14, 27);
    pempr_rec->name[27] = 0;
    /* print the name of the person to the name list file */
    printf(names, "%s\n", pempr_rec->name);
    strcpy(pempr_rec->pens_ind, ssarec+42, 1);
    pempr_rec->pens_ind[1] = 0;
    strcpy(pempr_rec->defcomp_ind, ssarec+44, 1);
    pempr_rec->defcomp_ind[1] = 0;
    strcpy(pempr_rec->wages, ssarec+47, 8);
    pempr_rec->wages[8] = 0;
    strcpy(pempr_rec->tips, ssarec+56, 8);
    pempr_rec->tips[8] = 0;
    strcpy(pempr_rec->other, ssarec+65, 10);
    pempr_rec->other[10] = 0;
    strcpy(pempr_rec->fed_tax, ssarec+76, 10);
    pempr_rec->fed_tax[10] = 0;
    strcpy(pempr_rec->fica_tax, ssarec+87, 7);
    pempr_rec->fica_tax[7] = 0;
    strcpy(pempr_rec->adv_earn_inc, ssarec+96, 8);
    pempr_rec->adv_earn_inc[8] = 0;
    strcpy(pempr_rec->med_wages, ssarec+105, 9);
    pempr_rec->med_wages[9] = 0;
    strcpy(pempr_rec->med_tax, ssarec+117, 7);
    pempr_rec->med_tax[7] = 0;
    strcpy(pempr_rec->ctrl_no, ssarec+125, 7);
    pempr_rec->ctrl_no[7] = 0;

    /* read and parse the second line of the employee record */
    fgets(ssarec, RECLEN, emplr);
    strcpy(pempr_rec->street_add, ssarec+14, 27);
    pempr_rec->street_add[27] = 0;
    strcpy(pempr_rec->dep_care, ssarec+53, 8);
    pempr_rec->dep_care[8] = 0;
    strcpy(pempr_rec->alloc_tips, ssarec+74, 8);
    pempr_rec->alloc_tips[8] = 0;
    strcpy(pempr_rec->grp_insur, ssarec+97, 8);
    pempr_rec->grp_insur[8] = 0;
    strcpy(pempr_rec->uncoll_fica_tax, ssarec+122, 8);
    pempr_rec->uncoll_fica_tax[8] = 0;

    /* read and parse the third line of the employee record */
    fgets(ssarec, RECLEN, emplr);
    strcpy(pempr_rec->city, ssarec+14, 18);
    pempr_rec->city[18] = 0;
    strcpy(pempr_rec->state, ssarec+33, 2);
    pempr_rec->state[2] = 0;
    strcpy(pempr_rec->zip_code, ssarec+36, 5);
    pempr_rec->zip_code[5] = 0;
}

```



```

    strncpy(pempl_rec->defcomp, ssarec+52, 10);
    pempl_rec->defcomp[10] = 0;
    strncpy(pempl_rec->sta, ssarec+68, 1);
    pempl_rec->sta[1] = 0;
    strncpy(pempl_rec->fr_ben, ssarec+78, 10);
    pempl_rec->fr_ben[10] = 0;
    strncpy(pempl_rec->nqsec, ssarec+99, 10);
    pempl_rec->nqsec[10] = 0;
    strncpy(pempl_rec->nqnot, ssarec+120, 10);
    pempl_rec->nqnot[10] = 0;
}

/*
 * parse_employer_record
 *
 * This function reads in the employer record, and parses the
 * data from the trash, and prints the record to the data file
 *
 * Input: Array containing first line of the record, ssarec.
 *         COM file pointer, emplr.
 *         array containing current mrn, curr_mrn.
 *         data structure for employer, pemplr_rec.
 *
 * Output: Data structure record is filled and current mrn
 *         is updated.
 */
parse_employer_rec(ssarec, emplr, curr_mrn, pemplr_rec)
char ssarec[], curr_mrn[];
FILE *emplr;
struct EAMATE_W2EMPLR_HEADER *pemplr_rec;
{
    /* parse the first line of the employer record */
    strncpy(pempl_rec->eln, ssarec+12, 10);
    pempl_rec->eln[10] = 0;
    strncpy(pempl_rec->est, ssarec+41, 4);
    pempl_rec->est[4] = 0;
    strncpy(pempl_rec->rpt_yr, ssarec+55, 4);
    pempl_rec->rpt_yr[4] = 0;
    strncpy(pempl_rec->proc_yr, ssarec+69, 4);
    pempl_rec->proc_yr[4] = 0;
    strncpy(pempl_rec->tape_lib_num, ssarec+87, 6);
    pempl_rec->tape_lib_num[6] = 0;
    strncpy(pempl_rec->type_emplr, ssarec+96, 1);
    pempl_rec->type_emplr[1] = 0;
    strncpy(pempl_rec->name_code, ssarec+98, 1);
    pempl_rec->name_code[1] = 0;
    strncpy(pempl_rec->other_elin, ssarec+111, 9);
    pempl_rec->other_elin[9] = 0;
    strncpy(pempl_rec->mrn, ssarec+121, 11);
    pempl_rec->mrn[11] = 0;
    strcpy(curr_mrn, pempl_rec->mrn);
    pempl_rec->final_offset = -1;
    pempl_rec->cum_offset = -1;

    /* read and parse the second employer line */
    fgets(ssarec, RECLEN, emplr);
    strncpy(pempl_rec->name, ssarec+2, 47);
    pempl_rec->name[47] = 0;
    strncpy(pempl_rec->street_add, ssarec+50, 40);
    pempl_rec->street_add[40] = 0;
    strncpy(pempl_rec->city, ssarec+91, 25);

```

```

)
/*
 * trash_record_labels
 *
 * this function reads in the employee labels and trashes
 * them.
 *
 * Input: pointer to the COM file, emplr.
 *
 * Output: employee labels are read in and trashed.
 */
trash_record_labels(emplr)
FILE *emplr;
{
    char recbuff[2*RECLEN];

    fgets(recbuff, 2*RECLEN-1, emplr);
}

/*
 * parse_interm_rec
 *
 * This function reads in the intermediate total, and parses the
 * data from the trash, and prints the record to the data file
 *
 * Input: Array containing first line of the record, ssarec.
 *
 * COM file pointer, emplr.
 *
 * data file pointer, data.
 *
 * Output: Data structure record is filled and printed to the
 *
 * data file.
 */
parse_interm_rec(ssarec, emplr, pempl_rec)
FILE *emplr;
char ssarec[];
struct EMATE_W2INTERMED_TOT *pempl_rec;
{
    char recbuff[2*RECLEN];

    /* Trash row of labels */
    fgets(recbuff, RECLEN, emplr);

    /* Read processed first line */
    fgets(ssarec, RECLEN, emplr);
    strncpy(pempl_rec->proc_wages, ssarec+17, 11);
    pempl_rec->proc_wages[11] = 0;
    strncpy(pempl_rec->proc_tips, ssarec+31, 11);
    pempl_rec->proc_tips[11] = 0;
    strncpy(pempl_rec->proc_other, ssarec+45, 11);
    pempl_rec->proc_other[11] = 0;
    strncpy(pempl_rec->proc_fed_tax, ssarec+58, 11);
    pempl_rec->proc_fed_tax[11] = 0;
    strncpy(pempl_rec->proc_fica_tax, ssarec+71, 11);
    pempl_rec->proc_fica_tax[11] = 0;
    strncpy(pempl_rec->proc_earn_inc, ssarec+84, 12);
    pempl_rec->proc_earn_inc[12] = 0;

    parse_total_record
    *
    * This function reads in the final total record, and parses the
    * data from the trash, and prints the record to the data file
    *
    * Input: Array containing first line of the record, ssarec.
    *
    * COM file pointer, emplr.
    *
    * data file pointer, data.
    *
    * Output: Data structure record is filled and printed to the
    *
    * data file.
    */
    parse_total_rec(ssarec, emplr, pempl_rec)
    char ssarec[];
    FILE *emplr;
    struct EMATE_W2FINAL_TOT *pempl_rec;

```

```

    }
    char recbuff(2*RECLEN);

    /* Trash row of labels */
    fgets(recbuff, RECLEN, emplr);

    /* Read processed first line */
    fgets(ssarec, RECLEN, emplr);
    strncpy(pempl_rec->proc_wages, ssarec+18, 14);
    pempl_rec->proc_wages[14] = 0;
    strncpy(pempl_rec->proc_tips, ssarec+35, 13);
    pempl_rec->proc_tips[13] = 0;
    strncpy(pempl_rec->proc_other, ssarec+51, 14);
    pempl_rec->proc_other[14] = 0;
    strncpy(pempl_rec->proc_fed_tax, ssarec+69, 13);
    pempl_rec->proc_fed_tax[13] = 0;
    strncpy(pempl_rec->proc_fica_tax, ssarec+86, 13);
    pempl_rec->proc_fica_tax[13] = 0;
    strncpy(pempl_rec->proc_earn_inc, ssarec+103, 13);
    pempl_rec->proc_earn_inc[13] = 0;
    strncpy(pempl_rec->proc_items, ssarec+118, 7);
    pempl_rec->proc_items[7] = 0;

    /* read reported first line */
    fgets(ssarec, RECLEN, emplr);
    strncpy(pempl_rec->rep_wages, ssarec+18, 14);
    pempl_rec->rep_wages[14] = 0;
    strncpy(pempl_rec->rep_tips, ssarec+35, 13);
    pempl_rec->rep_tips[13] = 0;
    strncpy(pempl_rec->rep_other, ssarec+51, 14);
    pempl_rec->rep_other[14] = 0;
    strncpy(pempl_rec->rep_fed_tax, ssarec+69, 13);
    pempl_rec->rep_fed_tax[13] = 0;
    strncpy(pempl_rec->rep_fica_tax, ssarec+86, 13);
    pempl_rec->rep_fica_tax[13] = 0;
    strncpy(pempl_rec->rep_earn_inc, ssarec+103, 13);
    pempl_rec->rep_earn_inc[13] = 0;
    strncpy(pempl_rec->rep_items, ssarec+118, 7);
    pempl_rec->rep_items[7] = 0;

    /* trash next two lines of labels */
    fgets(recbuff, 2*RECLEN-1, emplr);

    /* read processed second line */
    fgets(ssarec, RECLEN, emplr);
    strncpy(pempl_rec->proc_defcomp, ssarec+18, 14);
    pempl_rec->proc_defcomp[14] = 0;
    strncpy(pempl_rec->proc_nonqual, ssarec+34, 14);
    pempl_rec->proc_nonqual[14] = 0;
    strncpy(pempl_rec->proc_med_wages, ssarec+51, 14);
    pempl_rec->proc_med_wages[14] = 0;
    strncpy(pempl_rec->proc_med_tax, ssarec+69, 14);
    pempl_rec->proc_med_tax[14] = 0;

    /* read reported second line */
    fgets(ssarec, RECLEN, emplr);
    strncpy(pempl_rec->rep_defcomp, ssarec+18, 14);
    pempl_rec->rep_defcomp[14] = 0;
    strncpy(pempl_rec->rep_nonqual, ssarec+34, 14);
    pempl_rec->rep_nonqual[14] = 0;
    strncpy(pempl_rec->rep_med_wages, ssarec+51, 14);
    pempl_rec->rep_med_wages[14] = 0;
    strncpy(pempl_rec->rep_med_tax, ssarec+69, 14);
    pempl_rec->rep_med_tax[14] = 0;
}

/* parse_cum_ein
 *
 * This function reads in the cumulative ein total, and parses the
 * data from the trash, and prints the record to the data file
 *
 * Input: Array containing first line of the record, ssarec.
 *        com file pointer, emplr.
 *        data file pointer, data.
 *
 * Output: Data structure record is filled and printed to the
 *         data file.
 */
parse_cum_ein(ssarec, emplr, pempl_rec)
char *ssarec;
FILE *emplr;
struct EAMATE_W2CUMELN_TOT *pempl_rec;
{
    /* Read processed line */
    fgets(ssarec, RECLEN, emplr);
    strncpy(pempl_rec->proc_wages, ssarec+18, 14);
    pempl_rec->proc_wages[14] = 0;
    strncpy(pempl_rec->proc_tips, ssarec+35, 13);
    pempl_rec->proc_tips[13] = 0;
    strncpy(pempl_rec->proc_other, ssarec+51, 14);
    pempl_rec->proc_other[14] = 0;
    strncpy(pempl_rec->proc_fed_tax, ssarec+69, 13);
    pempl_rec->proc_fed_tax[13] = 0;
    strncpy(pempl_rec->proc_fica_tax, ssarec+86, 13);
    pempl_rec->proc_fica_tax[13] = 0;
    strncpy(pempl_rec->proc_earn_inc, ssarec+103, 13);
    pempl_rec->proc_earn_inc[13] = 0;
    strncpy(pempl_rec->proc_items, ssarec+118, 7);
    pempl_rec->proc_items[7] = 0;
}

```



```

# Bottom Level Makefile (~/.ssapilot/src/bin/search_addmatch)
#
# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependencies for the source files.
#
# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../..
LIBDIR      = $(PROJECT_ROOT)/lib
INCDIR      = $(PROJECT_ROOT)/include
BINDIR      = $(PROJECT_ROOT)/bin

# this is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE = search_addmatch
SRC         = search_addmatch.c
LIBS        = $(LIBDIR)/libgen_eamdate.a
CLIBS       = -lgen_eamdate -lm
OBJ         = search_addmatch.o
CFLAGS      = -I$(INCDIR) -L$(LIBDIR)
CC          = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependencies for each of the source files
depend : $(SRC)
    $(CC) -M $(CFLAGS) $(SRC) > dependlist
    sed -e '1,/^# DO NOT DELETE/Id' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv Makefile Makefile.bak
    mv mm.tmp Makefile
    rm -f dependlist

```

```

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
    $(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(BINDIR)

.c.o :
    $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
search_addmatch.o : search_addmatch.c
search_addmatch.o : /usr/include/stdio.h
search_addmatch.o : /usr/include/stdlib.h
search_addmatch.o : /usr/include/sys/types.h
search_addmatch.o : /usr/include/sys/select.h
search_addmatch.o : /usr/include/sys/time.h
search_addmatch.o : /usr/include/sys/types.h
search_addmatch.o : /usr/include/time.h
search_addmatch.o : /usr/include/sys/sysmacros.h
search_addmatch.o : /usr/include/sys/time.h
search_addmatch.o : /usr/include/malloc.h
search_addmatch.o : /usr/include/math.h
search_addmatch.o : /usr/include/floatpoint.h
search_addmatch.o : /usr/include/sys/leefp.h
search_addmatch.o : /usr/include/fcntl.h
search_addmatch.o : /usr/include/rpc/rpc.h
search_addmatch.o : /usr/include/rpc/types.h
search_addmatch.o : /usr/include/rpc/types.h
search_addmatch.o : /usr/include/sys/time.h
search_addmatch.o : /usr/include/tuser.h
search_addmatch.o : /usr/include/sys/tuser.h
search_addmatch.o : /usr/include/fcntl.h
search_addmatch.o : /usr/include/memory.h
search_addmatch.o : /usr/include/rpc/xdr.h
search_addmatch.o : /usr/include/sys/byteorder.h
search_addmatch.o : /usr/include/sys/types.h
search_addmatch.o : /usr/include/stdio.h
search_addmatch.o : /usr/include/rpc/auth.h
search_addmatch.o : /usr/include/rpc/xdr.h
search_addmatch.o : /usr/include/sys/cred.h
search_addmatch.o : /usr/include/sys/t_lock.h
search_addmatch.o : /usr/include/sys/machlock.h
search_addmatch.o : /usr/include/sys/types.h
search_addmatch.o : /usr/include/sys/dk_lkinfo.h
search_addmatch.o : /usr/include/sys/types.h
search_addmatch.o : /usr/include/sys/dl.h
search_addmatch.o : /usr/include/sys/sleepq.h
search_addmatch.o : /usr/include/sys/turnstile.h
search_addmatch.o : /usr/include/sys/types.h
search_addmatch.o : /usr/include/sys/param.h
search_addmatch.o : /usr/include/limits.h
search_addmatch.o : /usr/include/unistd.h
search_addmatch.o : /usr/include/sys/fcntl.h
search_addmatch.o : /usr/include/sys/signal.h
search_addmatch.o : /usr/include/vm/faultcode.h
search_addmatch.o : /usr/include/sys/types.h
search_addmatch.o : /usr/include/sys/pirec.h
search_addmatch.o : /usr/include/sys/sleepq.h
search_addmatch.o : /usr/include/sys/mutex.h
search_addmatch.o : /usr/include/sys/types.h
search_addmatch.o : /usr/include/sys/machlock.h
search_addmatch.o : /usr/include/sys/turnstile.h

```

```
search_addmatch.o: /usr/include/sys/dk1_lkinfo.h
search_addmatch.o: /usr/include/rpc/clnt.h
search_addmatch.o: /usr/include/rpc/rpc_com.h
search_addmatch.o: /usr/include/sys/netconfig.h
search_addmatch.o: /usr/include/rpc/rpc_msg.h
search_addmatch.o: /usr/include/rpc/clnt.h
search_addmatch.o: /usr/include/rpc/auth_sys.h
search_addmatch.o: /usr/include/rpc/auth_des.h
search_addmatch.o: /usr/include/rpc/auth_kerb.h
search_addmatch.o: /usr/include/kerberos/krb.h
search_addmatch.o: /usr/include/kerberos/mit-copyright.h
search_addmatch.o: /usr/include/kerberos/des.h
search_addmatch.o: /usr/include/sys/socket.h
search_addmatch.o: /usr/include/sys/netconfig.h
search_addmatch.o: /usr/include/netinet/in.h
search_addmatch.o: /usr/include/sys/stream.h
search_addmatch.o: /usr/include/sys/vnode.h
search_addmatch.o: /usr/ucbinclude/sys/types.h
search_addmatch.o: /usr/include/sys/t_lock.h
search_addmatch.o: /usr/include/sys/time.h
search_addmatch.o: /usr/include/sys/cred.h
search_addmatch.o: /usr/include/sys/uio.h
search_addmatch.o: /usr/ucbinclude/sys/types.h
search_addmatch.o: /usr/include/sys/poll.h
search_addmatch.o: /usr/include/sys/strmddep.h
search_addmatch.o: /usr/include/sys/cred.h
search_addmatch.o: /usr/include/sys/t_lock.h
search_addmatch.o: /usr/include/sys/byteorder.h
search_addmatch.o: /usr/include/rpc/svc.h
search_addmatch.o: /usr/include/rpc/rpc_com.h
search_addmatch.o: /usr/include/rpc/rpc_msg.h
search_addmatch.o: /usr/include/rpc/svc_auth.h
search_addmatch.o: /usr/include/rpc/svc.h
search_addmatch.o: /usr/include/rpc/rpcb_clnt.h
search_addmatch.o: /usr/include/rpc/types.h
search_addmatch.o: /usr/include/rpc/rpcb_prot.h
search_addmatch.o: /usr/include/rpc/rpc.h
search_addmatch.o: ../../include/params.h
search_addmatch.o: ../../include/eamestruct.h
search_addmatch.o: ../../include/btreestruct.h
```

```

/*
 * search_addmatch.c
 * version 4.0
 * 10/26/93
 *
 * by Natalie Willman
 *
 * This module contains the main search control routine to add
 * matches to a search request, and its auxiliary functions. It
 * must be linked with the record specific functions (eamate.c),
 * the general functions (general.c), and the btree functions (btrec.c).
 */

/* Include files */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/time.h>
#include <malloc.h>
#include <math.h>
#include <fcntl.h>
#include <rpc/rpc.h>
#include "params.h"
#include "eamstruct.h" /* Data file structure definitions */
#include "btrecstruct.h" /* B+ tree structure definitions */

/*
 * MAIN PROGRAM
 *
 * This program will register the rpc call and will enter a wait loop
 * to wait for a request.
 *
 * Listing of Functions and Prototypes:
 */
char *add_matches();

main()
{
    int rep;

    /* Register the RPC Call, and check for an error return */
    rep = register_rpc(0x37000000L, 1L, 1L, add_matches, xdr_int, xdr_int);
    if (rep == -1)
    {
        printf("ERROR: Cannot Register add_matches\n");
        perror("register_rpc(add_matches)");
    }

    /* Enter a wait loop for a user request */
    svc_run();

    /* SHOULD NEVER GET HERE!!!
    printf("ERROR: Returned from svc_run() in search_addmatch\n");
    perror("svc_run()");
    */

    /******
    /*
    /* Functions for add_matches()
    */
}

```

```

/*
 * This function accepts as input from the user a query structure
 * containing a year, ein and an index. It is called after a single
 * query request, and will print the next set (starting at the index)
 * of matches which were found by the search.
 */
/******
char *add_matches(user_num)
int *user_num;
{
    int ret_value, result;
    char namestring[FILENAME];
    char firstinitial[3];
    char filename[FILENAME], name[FILENAME];
    struct USER_QUERY user_query;
    FILE *fileptr, *brwptr;
    FILE *repptr, *listptr;
    long browse_offset, timer, timer1, statoffset;
    long *sortloc[DICE_GRAIN];
    int sortindex[DICE_GRAIN];
    int i, j, num, brwcount;
    time_t etimerst, etimerend, junk;

    printf("Additional Matches Request\n");

    /* Open the user query structure file
    sprintf(namestring, "query%d.txt", *user_num);
    fileptr = fopen(namestring, "rb");
    if (fileptr == NULL)
    {
        printf("ERROR: Cannot open file %s\n", namestring);
        ret_value = -1;
        return((char *) &ret_value);
    }

    /* Read in the user query structure
    if (fread(&user_query, sizeof(struct USER_QUERY), 1, fileptr) == 0)
    {
        printf("ERROR: Cannot read query\n");
        fclose(fileptr);
        ret_value = -1;
        return((char *) &ret_value);
    }
    fclose(fileptr);

    /* Print the user query structure -- for debugging information
    printf("User: %d\n", *user_num);
    printf("Year: %s\n", user_query.year);
    printf("EIN: %s\n", user_query.ein);
    printf("EST: %s\n", user_query.est);
    printf("offset: %s\n", user_query.offset);

    /* Open the output file to write the search results (offset/rank) to
    sprintf(namestring, "brw%d.txt", *user_num);
    brwptr = fopen(namestring, "w");
    if (brwptr == NULL)
    {
        printf("ERROR: Cannot open file %s\n", namestring);
        ret_value = -1;
        return((char *) &ret_value);
    }

```



```
    }

    /* Open the output file to write the search results (offset/rank) to */
    sprintf(namestring, "listoff%d.txt", *user_num);
    listptr = fopen(namestring, "r");
    if(listptr == NULL)
    {
        printf("ERROR: Cannot open file %s\n", namestring);
        fclose(brwptr);
        ret_value = -1;
        return((char *) &ret_value);
    }

    /* Open the output file to write the search results (offset/rank) to */
    /* open the report browse file */
    cr_browse_filename(namestring, user_query.year, "0", user_query.ein);
    reptr = fopen(namestring, "r");
    if(reptr == NULL)
    {
        printf("ERROR: Cannot open file %s\n", namestring);
        fclose(brwptr);
        fclose(listptr);
        ret_value = -1;
        return((char *) &ret_value);
    }

    /* Translate the offsets in the list of offsets to browse data */
    ret_value = translate_offsets(listptr, brwptr, reptr, atol(user_query.offset));

    /* Close all of the files and return a value for success/fail */
    fclose(listptr);
    fclose(brwptr);
    fclose(reptr);
    return((char *) &ret_value);
}
```

```

# Bottom Level Makefile (~/.ssapilot/src/bin/search_blanket)

# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependencies for the source files.

# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../../..
LIBDIR       = $(PROJECT_ROOT)/lib
INCDIR       = $(PROJECT_ROOT)/include
BINDIR       = $(PROJECT_ROOT)/bin

# this is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE = search_blanket
SRC         = search_blanket.c
LIBS        = $(LIBDIR)/libgen_eamte.a
CLIBS       = -lgen_eamte -lm
OBJ         = search_blanket.o
CFLAGS      = -I$(INCDIR) -L$(LIBDIR)
CC          = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependencies for each of the source files
depend : $(SRC)
    $(CC) -M $(CFLAGS) $(SRC) > dependlist
    sed -e '1,/"# DO NOT DELETE/id' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv Makefile Makefile.bak
    mv mm.tmp Makefile
    rm -f dependlist

```

```

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
    $(CC) $(OBJ) $(LIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(BINDIR)

.C.o :
    $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
search_blanket.o : search_blanket.c
search_blanket.o : /usr/include/stdio.h
search_blanket.o : /usr/include/stdlib.h
search_blanket.o : /usr/include/sys/types.h
search_blanket.o : /usr/include/sys/select.h
search_blanket.o : /usr/include/sys/time.h
search_blanket.o : /usr/include/sys/types.h
search_blanket.o : /usr/include/time.h
search_blanket.o : /usr/include/sys/sysmacros.h
search_blanket.o : /usr/include/sys/time.h
search_blanket.o : /usr/include/malloc.h
search_blanket.o : /usr/include/math.h
search_blanket.o : /usr/include/floatpoint.h
search_blanket.o : /usr/include/sys/leefp.h
search_blanket.o : /usr/include/fcntl.h
search_blanket.o : /usr/include/rpc/rpc.h
search_blanket.o : /usr/include/rpc/types.h
search_blanket.o : /usr/include/sys/types.h
search_blanket.o : /usr/include/sys/time.h
search_blanket.o : /usr/include/tuser.h
search_blanket.o : /usr/include/sys/tuser.h
search_blanket.o : /usr/include/memory.h
search_blanket.o : /usr/include/rpc/xdr.h
search_blanket.o : /usr/include/sys/byteorder.h
search_blanket.o : /usr/include/sys/types.h
search_blanket.o : /usr/include/stdio.h
search_blanket.o : /usr/include/rpc/auth.h
search_blanket.o : /usr/include/rpc/xdr.h
search_blanket.o : /usr/include/sys/cred.h
search_blanket.o : /usr/include/sys/t_lock.h
search_blanket.o : /usr/include/sys/machlock.h
search_blanket.o : /usr/include/sys/types.h
search_blanket.o : /usr/include/sys/dkinfo.h
search_blanket.o : /usr/include/sys/types.h
search_blanket.o : /usr/include/sys/dl.h
search_blanket.o : /usr/include/sys/sleepq.h
search_blanket.o : /usr/include/sys/turnstile.h
search_blanket.o : /usr/include/sys/types.h
search_blanket.o : /usr/include/sys/param.h
search_blanket.o : /usr/include/limits.h
search_blanket.o : /usr/include/unistd.h
search_blanket.o : /usr/include/sys/fcntl.h
search_blanket.o : /usr/include/sys/signal.h
search_blanket.o : /usr/include/vm/faultcode.h
search_blanket.o : /usr/include/sys/types.h
search_blanket.o : /usr/include/sys/pirec.h
search_blanket.o : /usr/include/sys/sleepq.h
search_blanket.o : /usr/include/sys/mutex.h
search_blanket.o : /usr/include/sys/types.h
search_blanket.o : /usr/include/sys/machlock.h
search_blanket.o : /usr/include/sys/turnstile.h

```

```
search_blanket.o: /usr/include/sys/dk1_kinfo.h
search_blanket.o: /usr/include/rpc/cint.h
search_blanket.o: /usr/include/rpc/rpc_com.h
search_blanket.o: /usr/include/sys/netconfig.h
search_blanket.o: /usr/include/rpc/rpc_msg.h
search_blanket.o: /usr/include/rpc/cint.h
search_blanket.o: /usr/include/rpc/auth_sys.h
search_blanket.o: /usr/include/rpc/auth_des.h
search_blanket.o: /usr/include/rpc/auth_kerb.h
search_blanket.o: /usr/include/kerberos/krb.h
search_blanket.o: /usr/include/kerberos/mit-copyright.h
search_blanket.o: /usr/include/kerberos/des.h
search_blanket.o: /usr/include/kerberos/mit-copyright.h
search_blanket.o: /usr/include/sys/socket.h
search_blanket.o: /usr/include/sys/netconfig.h
search_blanket.o: /usr/include/netinet/in.h
search_blanket.o: /usr/include/sys/stream.h
search_blanket.o: /usr/include/sys/vnode.h
search_blanket.o: /usr/include/sys/types.h
search_blanket.o: /usr/include/sys/t_lock.h
search_blanket.o: /usr/include/sys/time.h
search_blanket.o: /usr/include/sys/cred.h
search_blanket.o: /usr/include/sys/uio.h
search_blanket.o: /usr/include/sys/types.h
search_blanket.o: /usr/include/sys/poll.h
search_blanket.o: /usr/include/sys/strmdep.h
search_blanket.o: /usr/include/sys/cred.h
search_blanket.o: /usr/include/sys/t_lock.h
search_blanket.o: /usr/include/sys/byteorder.h
search_blanket.o: /usr/include/rpc/svc.h
search_blanket.o: /usr/include/rpc/rpc_com.h
search_blanket.o: /usr/include/rpc/rpc_msg.h
search_blanket.o: /usr/include/rpc/svc_auth.h
search_blanket.o: /usr/include/rpc/svc.h
search_blanket.o: /usr/include/rpc/rpcb_cint.h
search_blanket.o: /usr/include/rpc/types.h
search_blanket.o: /usr/include/rpc/rpcb_prot.h
search_blanket.o: /usr/include/rpc/rpc.h
search_blanket.o: ../../include/params.h
search_blanket.o: ../../include/eamateststruct.h
search_blanket.o: ../../include/btreestruct.h
```



```

/*
 * search_blanket.c
 * version 4.0
 * 10/26/93
 * by Natalie Willman
 *
 * This module contains the main search control routine for the
 * blanket request, and its auxiliary functions.
 */

/* Include files */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/time.h>
#include <malloc.h>
#include <math.h>
#include <fcntl.h>
#include <rpc/rpc.h>
#include "params.h"
#include "emmatestruct.h" /* Data file structure definitions */
#include "btreestruct.h" /* B+ tree structure definitions */

/*
 * MAIN PROGRAM
 *
 * This program will register the rpc call and will enter a wait
 * loop for requests.
 *
 * Listing of Functions and Prototypes:
 */

char *get_blanket();

main()
{
    int rep;

    /* register the RPC call and check for an error return */
    rep = register_rpc(0x3200000L, 1L, 1L, get_blanket, xdr_int, xdr_int);
    if (rep == -1)
    {
        printf("ERROR: Cannot register search_blanket\n");
        perror("register_rpc(get_blanket)");
    }

    /* Enter a wait loop and wait for user requests */
    svc_run();

    /* SHOULD NEVER REACH HERE!!!
    printf("ERROR: returned from svc_run() in search_blanket\n");
    perror("svc_run()");
    */

    /* Functions for get_blanket() */
    /* Print the query information -- for debugging purposes
    printf("User: %d\n", *user_num);
    */
}

/*
 * This function accepts as input from the user a query structure
 * containing a year, ein and sequence number. It opens the employee
 * detail file and pulls a certain number of records from the beginning
 * middle and end of the report and prints the record info to an output
 * file.
 */
/*
 * *****
 */
/*
 * get_blanket
 *
 * This function reads a user query and will generate
 * a blanket file for the user, consisting of 30 records,
 * 10 from the beginning, 10 from the middle, and 10 from the
 * end of the file.
 *
 * Input: User number
 *
 * Output: status of the function
 */
char *get_blanket(user_num)
int *user_num;
{
    int ret_value;
    char namestring[FILENAME], typed;
    struct USER_QUERY user_query;
    FILE *fileptr, *blanketptr, *headerptr;
    FILE *detailptr, *browseptr;
    long browse_offset, start_offset, mid_offset, end_offset;
    struct EMATE_W2EMPLR INFO employer_info;
    struct EMATE_W2EMPLR rec[BLANKET_SIZE];
    struct EMATE_W2EMPLR detail_rec;
    int i;

    printf("Blanket Request\n");

    /* Open the user's query structure
    sprintf(namestring, "query%d.txt", *user_num);
    fileptr = fopen(namestring, "rb");
    if (fileptr == NULL)
    {
        printf("ERROR: Cannot open file %s\n", namestring);
        ret_value = -1;
        return((char *) &ret_value);
    }

    /* Read the user's query structure
    if (fread(&user_query, sizeof(struct USER_QUERY), 1, fileptr) == 0)
    {
        printf("ERROR: Cannot read query\n");
        fclose(fileptr);
        ret_value = -1;
        return((char *) &ret_value);
    }
    fclose(fileptr);
    */
}

```

```

printf("Year:  %s\n", user_query.year);
printf("EIN:   %s\n", user_query.ein);
printf("EST:   %s\n", user_query.est);
printf("SEQ_NO: %s\n", user_query.seq_no);

/* Open the output file -- to write the user output data
   sprintf(filename, "blank%d.txt", *user_num);
   blanketptr = fopen(filename, "w");
   If (blanketptr == NULL)
   {
       printf("ERROR: Cannot open file %s\n", filename);
       ret_value = -1;
       return((char *) &ret_value);
   }

/* Open the output file -- to write the user output data
   sprintf(filename, "hdr%d.txt", *user_num);
   headerptr = fopen(filename, "w");
   If (headerptr == NULL)
   {
       printf("ERROR: Cannot open file %s\n", filename);
       ret_value = -1;
       return((char *) &ret_value);
   }

/* search for the employer header record for the user query
   browse_offset = search_seq_emplr(user_query.year, user_query.ein,
                                   user_query.seq_no, headerptr);

/* If the report exists
   If (browse_offset != -1)
   {
       /* read the header information and store it in a structure
       fseek(headerptr, 0, 0);
       read_eamate_w2header_info(headerptr, &employer_info);
       fclose(headerptr);

/* Open the browse file
       cr_browse_filename(filename, user_query.year, "0", user_query.ein);
       browseptr = fopen(filename, "r");
       If (browseptr == NULL)
       {
           fclose(blanketptr);
           printf("ERROR: Cannot open file %s\n", filename);
           ret_value = -1;
           return((char *) &ret_value);
       }

/* calculate the offsets for the records to be put in the blanket
       start_offset = sizeof(struct EAMATE_W2EMPL_BRW) *
                       employer_info.browse_start;
       end_offset = ( employer_info.num_recs-1) *
                    sizeof(struct EAMATE_W2EMPL_BRW) + start_offset;
       mid_offset = ( (long) employer_info.num_recs/2) *
                    sizeof(struct EAMATE_W2EMPL_BRW) + start_offset;

/* read the records from the beginning of the report
       fseek(browseptr, start_offset, 0);
       fread(rec, sizeof(struct EAMATE_W2EMPL_BRW), BLANKETFACTOR, browseptr);

/* read the records from the middle of the report
       fseek(browseptr, mid_offset, 0);
       fread(&rec(BLANKETFACTOR), sizeof(struct EAMATE_W2EMPL_BRW),
            BLANKETFACTOR, browseptr);

```

```

/* read the records from the end of the report
       fseek(browseptr, end_offset, 0);
       fread(&rec(BLANKETFACTOR*2), sizeof(struct EAMATE_W2EMPL_BRW),
            BLANKETFACTOR, browseptr);

/* create the detail filename and open the detail file
       cr_detail_filename(filename, user_query.year, user_query.ein,
                           user_query.seq_no, employer_info.platter_side);
       detailptr = fopen(filename, "r");
       If (detailptr == NULL)
       {
           fclose(blanketptr);
           fclose(browseptr);
           printf("ERROR: Cannot open file %s\n", filename);
           ret_value = -1;
           return((char *) &ret_value);
       }

/* for each of the records in the blanket, read in the detail data
/* and print it to the output file
   for (i = 0; i < BLANKETSIZE; i++)
   {
       fseek(detailptr, rec[i].record_loc, 0);
       fread(&typeld, sizeof(char), 1, detailptr);
       fread(&detail_rec, sizeof(struct EAMATE_W2EMPL_DETAIL), 1, detailptr);
       fwrite(&detail_rec, sizeof(struct EAMATE_W2EMPL_DETAIL), 1, blanketptr);
   }

/* close the file pointers and return a value of success
   fclose(browseptr);
   fclose(detailptr);
   fclose(blanketptr);
   ret_value = 1;
   return((char *) &ret_value);
   }
   else /* report does not exist
   {
       /* close the file pointers and return a value of fail
       fclose(blanketptr);
       ret_value = -1;
       return((char *) &ret_value);
   }
}

```

```

# Bottom Level Makefile (-/ssaplot/src/bin/search_browse)

# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependancies for the source files.

# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../../..
LIBDIR      = $(PROJECT_ROOT)/lib
INCDIR      = $(PROJECT_ROOT)/include
BINDIR      = $(PROJECT_ROOT)/bin

# this is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE  = search_browse
SRC          = search_browse.c
LIBS         = $(LIBDIR)/libgen_eamate.a
CLIBS        = -lgen_eamate -lm
OBJ           = search_browse.o
CFLAGS       = -I$(INCDIR) -I$(LIBDIR)
CC           = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependancies for each of the source files
depend : $(SRC)
    $(CC) -M $(CFLAGS) $(SRC) > dependlist
    sed -e '1,/^# DO NOT DELETE/d' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv Makefile Makefile.bak
    mv mm.tmp Makefile
    rm -f dependlist

```

```

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
    $(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(BINDIR)

.c.o :
    $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
search_browse.o: search_browse.c
search_browse.o: /usr/ucbinclude/stdio.h
search_browse.o: /usr/include/stdlib.h
search_browse.o: /usr/ucbinclude/sys/types.h
search_browse.o: /usr/include/sys/select.h
search_browse.o: /usr/include/sys/time.h
search_browse.o: /usr/ucbinclude/sys/types.h
search_browse.o: /usr/include/time.h
search_browse.o: /usr/ucbinclude/sys/sysmacros.h
search_browse.o: /usr/include/sys/time.h
search_browse.o: /usr/include/malloc.h
search_browse.o: /usr/include/math.h
search_browse.o: /usr/include/floatnpoint.h
search_browse.o: /usr/include/sys/ieeeep.h
search_browse.o: /usr/ucbinclude/fcntl.h
search_browse.o: /usr/include/rpc/rpc.h
search_browse.o: /usr/include/rpc/types.h
search_browse.o: /usr/ucbinclude/sys/types.h
search_browse.o: /usr/include/sys/time.h
search_browse.o: /usr/include/tuser.h
search_browse.o: /usr/ucbinclude/fcntl.h
search_browse.o: /usr/include/memory.h
search_browse.o: /usr/include/rpc/xdr.h
search_browse.o: /usr/include/sys/byteorder.h
search_browse.o: /usr/ucbinclude/sys/types.h
search_browse.o: /usr/ucbinclude/stdio.h
search_browse.o: /usr/include/rpc/auth.h
search_browse.o: /usr/include/rpc/xdr.h
search_browse.o: /usr/include/sys/cred.h
search_browse.o: /usr/include/sys/t_lock.h
search_browse.o: /usr/include/sys/machlock.h
search_browse.o: /usr/ucbinclude/sys/types.h
search_browse.o: /usr/include/sys/dki_kinfo.h
search_browse.o: /usr/ucbinclude/sys/types.h
search_browse.o: /usr/include/sys/sleepq.h
search_browse.o: /usr/include/sys/turnstile.h
search_browse.o: /usr/ucbinclude/sys/types.h
search_browse.o: /usr/ucbinclude/sys/param.h
search_browse.o: /usr/include/limits.h
search_browse.o: /usr/ucbinclude/unistd.h
search_browse.o: /usr/ucbinclude/sys/fcntl.h
search_browse.o: /usr/ucbinclude/sys/signal.h
search_browse.o: /usr/include/vm/faultcode.h
search_browse.o: /usr/ucbinclude/sys/types.h
search_browse.o: /usr/ucbinclude/sys/pirec.h
search_browse.o: /usr/include/sys/sleepq.h
search_browse.o: /usr/include/sys/mutex.h
search_browse.o: /usr/ucbinclude/sys/types.h
search_browse.o: /usr/ucbinclude/sys/machlock.h
search_browse.o: /usr/include/sys/turnstile.h

```



```
search_browse.o: /usr/include/sys/dkinfo.h
search_browse.o: /usr/include/rpc/clnt.h
search_browse.o: /usr/include/rpc/rpc_com.h
search_browse.o: /usr/include/sys/netconfig.h
search_browse.o: /usr/include/rpc/rpc_msg.h
search_browse.o: /usr/include/rpc/clnt.h
search_browse.o: /usr/include/rpc/auth_sys.h
search_browse.o: /usr/include/rpc/auth_des.h
search_browse.o: /usr/include/rpc/auth_kerb.h
search_browse.o: /usr/include/kerberos/krb.h
search_browse.o: /usr/include/kerberos/mit-copyright.h
search_browse.o: /usr/include/kerberos/des.h
search_browse.o: /usr/include/kerberos/mit-copyright.h
search_browse.o: /usr/include/sys/socket.h
search_browse.o: /usr/include/sys/netconfig.h
search_browse.o: /usr/include/netinet/in.h
search_browse.o: /usr/include/sys/stream.h
search_browse.o: /usr/include/sys/vnode.h
search_browse.o: /usr/include/sys/types.h
search_browse.o: /usr/include/sys/t_lock.h
search_browse.o: /usr/include/sys/time.h
search_browse.o: /usr/include/sys/cred.h
search_browse.o: /usr/include/sys/uio.h
search_browse.o: /usr/include/sys/types.h
search_browse.o: /usr/include/sys/poll.h
search_browse.o: /usr/include/sys/stream.h
search_browse.o: /usr/include/sys/cred.h
search_browse.o: /usr/include/sys/t_lock.h
search_browse.o: /usr/include/sys/byteorder.h
search_browse.o: /usr/include/rpc/svc.h
search_browse.o: /usr/include/rpc/rpc_com.h
search_browse.o: /usr/include/rpc/rpc_msg.h
search_browse.o: /usr/include/rpc/svc_auth.h
search_browse.o: /usr/include/rpc/svc.h
search_browse.o: /usr/include/rpc/rpcb_clnt.h
search_browse.o: /usr/include/rpc/types.h
search_browse.o: /usr/include/rpc/rpcb_prot.h
search_browse.o: /usr/include/rpc/rpc.h
search_browse.o: ../../include/param.h
search_browse.o: ../../include/unistd.h
search_browse.o: ../../include/btreetruct.h
```

```

/*
 * search_browse.c
 * version 4.0
 * 10/26/93
 *
 * by Natalie Willman
 *
 * This module contains the main search control routine for browse
 * report, and its auxiliary functions.
 */

/* Include Files */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/time.h>
#include <malloc.h>
#include <math.h>
#include <fcntl.h>
#include <rpc/rpc.h>
#include <params.h>
#include "eamstruct.h" /* Data file structure definitions */
#include "btreestruct.h" /* B+ tree structure definitions */

/*
 * MAIN PROGRAM
 *
 * This program will register the rpc call and will enter a wait
 * loop for requests.
 *
 * Listing of functions and Prototypes:
 */
char *browse_report();

main()
{
    int rep;

    /* register the rpc call and check for an error return
     rep = register_rpc(0x35000000L, 1L, 1L, browse_report, xdr_int, xdr_int);
     if (rep == -1)
     {
         printf("ERROR: Cannot register browse_report\n");
         perror("register_rpc(browse_report)");
     }

     /* enter a wait loop to wait for user requests
     svc_run();

     /* SHOULD NEVER REACH HERE!!!
     printf("ERROR: returned from svc_run() in search_browse\n");
     perror("svc_run()");
     }

     *****
     /* Functions for browse_report()
     */
}

/* This function accepts as input from the user a query structure
 * containing a year, ein and microfilm ref number. It searches the
 * employee browse file for the first record with this mrn and prints
 * the offset to it to an output file.
 */
/*****
char *browse_report(user_num)
int *user_num;
{
    int ret_value, num, i;
    char namestring[FILENAME], seq_no[SEQ_SIZE+1];
    struct USER_QUERY user_query;
    struct EAMFILE W2EMPL_BRM brw;
    FILE *fileptr, *browserptr, *headerptr;
    long browse_offset;

    printf("Browse Report Request\n");

    /* Open the user query structure file
    sprintf(namestring, "query%d.txt", *user_num);
    fileptr = fopen(namestring, "rb");
    if (fileptr == NULL)
    {
        printf("ERROR: Cannot open file %s\n", namestring);
        ret_value = -1;
        return((char *) &ret_value);
    }

    /* Read the user query structure
    if (fread(&user_query, sizeof(struct USER_QUERY), 1, fileptr) == 0)
    {
        printf("ERROR: Cannot read query\n");
        fclose(fileptr);
        ret_value = -1;
        return((char *) &ret_value);
    }

    /* print the user query information -- for debugging purposes
    printf("User: %d\n", *user_num);
    printf("Year: %s\n", user_query.year);
    printf("EIN: %s\n", user_query.ein);
    printf("EST: %s\n", user_query.est);
    printf("MRN: %s\n", user_query.offset);
    fclose(fileptr);

    /* Open the Output file to place the browse records into
    sprintf(namestring, "brow%d.txt", *user_num);
    fileptr = fopen(namestring, "w");
    if (fileptr == NULL)
    {
        printf("ERROR: Cannot open file %s\n", namestring);
        ret_value = -1;
        return((char *) &ret_value);
    }

    /* Open the Output file to place the employer header info into
    headerptr = fopen(namestring, "w");
    if (headerptr == NULL)
    {
        printf("ERROR: Cannot open file %s\n", namestring);
        ret_value = -1;
    }
}
/*****

```

```

return((char *) &ret_value);
}

/* search for the start of the report by mrn
browse_offset = search_mrn(user_query.ein, user_query.year,
user_query.offset);
if(browse_offset != -1)
{
printf("browse offset %d\n", browse_offset);
cr_browse_filename(namestring, user_query.year, "0", user_query.ein);
browseptr = fopen(namestring, "r");
if(browseptr == NULL)
{
/* return an unsuccessful result
fclose(fileptr);
fclose(headerptr);
printf("ERROR: Cannot open file %s\n", namestring);
ret_value = -1;
return((char *) &ret_value);
}

/* write the browse recs to a file and return a successful result
fseek(browseptr, browse_offset, 0);
num = fread(&brw, sizeof(struct EAMATE_W2EMPL_BRW), 1, browseptr);
if (num != 0)
strcpy(seq_no, brw.seq_no);
for(i = 1; i < BROWSENUM; i++)
{
num = fread(&brw, sizeof(struct EAMATE_W2EMPL_BRW), 1, browseptr);
if (num != 0)
{write(&brw, sizeof(struct EAMATE_W2EMPL_BRW), 1, fileptr);
}
}

search_seq_empty(user_query.year, user_query.ein, seq_no, headerptr);
fclose(headerptr);
fclose(fileptr);
fclose(browseptr);
ret_value = 1;
return((char *) &ret_value);
}
else
{
/* return an unsuccessful result
fclose(fileptr);
fclose(headerptr);
printf("ERROR: Unable to browse report\n");
ret_value = -1;
return((char *) &ret_value);
}
}

```



```

# Bottom Level Makefile (-/ssapilot/src/bin/search_detail)

# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependancies for the source files.

# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../../..
LIBDIR      = $(PROJECT_ROOT)/lib
INCDIR      = $(PROJECT_ROOT)/include
BINDIR      = $(PROJECT_ROOT)/bin

# this is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE  = search_detail
SRC          = search_detail.c
LIBS         = $(LIBDIR)/libgen eamdate.a
CLIBS        = -lgen_eamdate -lm
OBJ          = search_detail.o
CFLAGS       = -IS(INCDIR) -I$(LIBDIR)
CC           = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean : rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
      rm -f $(EXECUTABLE)
      rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependancies for each of the source files
depend : $(SRC)
      sed -e '1,/^# DO NOT DELETE/id' Makefile > mm.tmp
      cat dependlist >> mm.tmp
      mv Makefile Makefile.bak
      mv mm.tmp Makefile
      rm -f dependlist

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
      $(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
      cp $(EXECUTABLE) $(BINDIR)

.c.o : $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
search_detail.o: search_detail.c
search_detail.o: /usr/include/stdio.h
search_detail.o: /usr/include/stdlib.h
search_detail.o: /usr/ucbinclude/sys/types.h
search_detail.o: /usr/include/sys/select.h
search_detail.o: /usr/include/sys/time.h
search_detail.o: /usr/ucbinclude/sys/types.h
search_detail.o: /usr/include/time.h
search_detail.o: /usr/ucbinclude/sys/symacros.h
search_detail.o: /usr/include/sys/time.h
search_detail.o: /usr/include/math.h
search_detail.o: /usr/include/floatnpoint.h
search_detail.o: /usr/include/sys/ieeefp.h
search_detail.o: /usr/ucbinclude/fcntl.h
search_detail.o: /usr/include/rpc/rpc.h
search_detail.o: /usr/include/rpc/types.h
search_detail.o: /usr/ucbinclude/sys/types.h
search_detail.o: /usr/include/sys/time.h
search_detail.o: /usr/include/tuser.h
search_detail.o: /usr/include/sys/tuser.h
search_detail.o: /usr/ucbinclude/fcntl.h
search_detail.o: /usr/include/memory.h
search_detail.o: /usr/include/rpc/xdr.h
search_detail.o: /usr/include/sys/byteorder.h
search_detail.o: /usr/ucbinclude/sys/types.h
search_detail.o: /usr/ucbinclude/stdio.h
search_detail.o: /usr/include/rpc/auth.h
search_detail.o: /usr/include/rpc/xdr.h
search_detail.o: /usr/include/sys/cred.h
search_detail.o: /usr/include/sys/t_lock.h
search_detail.o: /usr/include/sys/machlock.h
search_detail.o: /usr/ucbinclude/sys/types.h
search_detail.o: /usr/include/sys/dk1_kinfo.h
search_detail.o: /usr/ucbinclude/sys/types.h
search_detail.o: /usr/include/sys/dl.h
search_detail.o: /usr/include/sys/sleepq.h
search_detail.o: /usr/include/sys/turnstile.h
search_detail.o: /usr/ucbinclude/sys/types.h
search_detail.o: /usr/ucbinclude/sys/param.h
search_detail.o: /usr/ucbinclude/limits.h
search_detail.o: /usr/ucbinclude/unistd.h
search_detail.o: /usr/ucbinclude/sys/fcntl.h
search_detail.o: /usr/ucbinclude/sys/signal.h
search_detail.o: /usr/include/vm/faultcode.h
search_detail.o: /usr/ucbinclude/sys/types.h
search_detail.o: /usr/include/sys/pirec.h
search_detail.o: /usr/include/sys/sleepq.h
search_detail.o: /usr/include/sys/mutex.h
search_detail.o: /usr/ucbinclude/sys/types.h
search_detail.o: /usr/include/sys/machlock.h
search_detail.o: /usr/include/sys/turnstile.h

```

```
search_detail.o: /usr/include/sys/dk1_lkinfo.h
search_detail.o: /usr/include/rpc/clnt.h
search_detail.o: /usr/include/rpc/rpc_com.h
search_detail.o: /usr/include/sys/netconfig.h
search_detail.o: /usr/include/rpc/rpc_msg.h
search_detail.o: /usr/include/rpc/clnt.h
search_detail.o: /usr/include/rpc/auth_sys.h
search_detail.o: /usr/include/rpc/auth_des.h
search_detail.o: /usr/include/rpc/auth_kerb.h
search_detail.o: /usr/include/kerberos/krb.h
search_detail.o: /usr/include/kerberos/mit-copyright.h
search_detail.o: /usr/include/kerberos/des.h
search_detail.o: /usr/include/sys/socket.h
search_detail.o: /usr/include/sys/netconfig.h
search_detail.o: /usr/include/netinet/in.h
search_detail.o: /usr/include/sys/stream.h
search_detail.o: /usr/include/sys/vnode.h
search_detail.o: /usr/ucbinclude/sys/types.h
search_detail.o: /usr/include/sys/t_lock.h
search_detail.o: /usr/include/sys/time.h
search_detail.o: /usr/include/sys/cred.h
search_detail.o: /usr/include/sys/uio.h
search_detail.o: /usr/ucbinclude/sys/types.h
search_detail.o: /usr/ucbinclude/sys/poll.h
search_detail.o: /usr/include/sys/strmdep.h
search_detail.o: /usr/include/sys/cred.h
search_detail.o: /usr/include/sys/t_lock.h
search_detail.o: /usr/include/sys/byteorder.h
search_detail.o: /usr/include/rpc/svc.h
search_detail.o: /usr/include/rpc/rpc_com.h
search_detail.o: /usr/include/rpc/rpc_msg.h
search_detail.o: /usr/include/rpc/svc_auth.h
search_detail.o: /usr/include/rpc/svc.h
search_detail.o: /usr/include/rpc/rpcb_clnt.h
search_detail.o: /usr/include/rpc/types.h
search_detail.o: /usr/include/rpc/rpcb_prot.h
search_detail.o: /usr/include/rpc/rpc.h
search_detail.o: ../../include/eamatestruct.h
search_detail.o: ../../include/btreestruct.h
```

```

/*
 * search_detail.c
 * version 4.0
 * 10/26/93
 *
 * by Natalie Willman
 *
 * This module contains the main search control routine for the
 * get employee detail request, and its auxiliary functions.
 */

/* Include Files */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/time.h>
#include <malloc.h>
#include <math.h>
#include <fcntl.h>
#include <rpc/rpc.h>
#include <params.h>
#include "eamteststruct.h" /* Data file structure definitions */
#include "btroestruct.h" /* B+ tree structure definitions */

/*
 * MAIN PROGRAM
 */
/* This program will register the rpc call and will enter a wait loop
 * waiting for requests.
 */
/* Listing of Functions and Prototypes:
 */
char *get_empl_detail();

main()
{
    int rep;

    /* Register the rpc call and check for an error return
    rep = register_rpc(0x34000000L, 1L, 1L, get_empl_detail, xdr_int, xdr_int);
    if (rep == -1)
    {
        printf("ERROR: Cannot register search_detail\n");
        perror("register_rpc(get_empl_detail)");
    }

    /* Enter a wait loop waiting for a request from the user
    svc_run();

    /* SHOULD NEVER REACH HERE
    printf("ERROR: returned from svc_run() in search_detail\n");
    perror("svc_run()");
    }

    /* Functions for get_empl_detail()
    */
    /* This function accepts as input from the user a query structure
    */
    containing a year, eln and sequence number, and offset. It searches */
    /* the employee detail file for the offset and prints the record info */
    /* to an output file.
    */
    /*
    *****
    char *get_empl_detail(user_num)
    int *user_num;
    {
        int ret_value;
        char namestring[FILENAME], typeid;
        struct USER_QUERY user_query;
        FILE *fileptr, *emlptr;
        long browse_offset;
        struct FAXATE_W2EMPL_DETAIL employee_detail;
        struct FAXATE_W2EMPLR INFO employer_info;

        printf("Employee Detail Request\n");

        /* Open the user query structure file
        sprintf(namestring, "query%d.txt", *user_num);
        fileptr = fopen(namestring, "rb");
        if (fileptr == NULL)
        {
            printf("ERROR: Cannot open file %s\n", namestring);
            ret_value = -1;
            return((char *) &ret_value);
        }

        /* Read in the user query
        if (fread(&user_query, sizeof(struct USER_QUERY), 1, fileptr) == 0)
        {
            printf("ERROR: Cannot read query\n");
            fclose(fileptr);
            ret_value = -1;
            return((char *) &ret_value);
        }

        /* Print the user query information -- for debugging purposes
        printf("User: %d\n", *user_num);
        printf("Year: %s\n", user_query.Year);
        printf("EIN: %s\n", user_query.eln);
        printf("EST: %s\n", user_query.est);
        printf("SEQ: %s\n", user_query.seq_no);
        printf("OFFSET: %s\n", user_query.offset);
        fclose(fileptr);

        /* Open the file to which to print the output
        sprintf(namestring, "detail%d.txt", *user_num);
        fileptr = fopen(namestring, "w+");
        if (fileptr == NULL)
        {
            printf("ERROR: Cannot open file %s\n", namestring);
            ret_value = -1;
            return((char *) &ret_value);
        }

        /* Check to see if the employer report exists
        browse_offset = search_seq_empl(user_query.Year, user_query.eln,
        user_query.seq_no, fileptr);

```



```
if(browse_offset != -1)
{
    /* read in and store the employer header information */
    fseek(fileptr, 0, 0);
    read_employee_header_info(fileptr, &employer_info);
    fclose(fileptr);
    fileptr = fopen(namestring, "w");

    /* create and open the employer report detail file
    cr_detail_filename(namestring, user_query.year, user_query.eid,
        user_query.seq_no, employer_info.platter_side);
    emplptr = fopen(namestring, "r");
    if(emplptr == NULL)
    {
        printf("ERROR: Cannot open file %s\n", namestring);
        fclose(fileptr);
        ret_value = -1;
        return((char *) &ret_value);
    }
    else
    {
        /* seek to the detail offset and read in the typeid
        fseek(emplptr, atol(user_query.offset), 0);
        if(fread(&typeid, sizeof(char), 1, emplptr) == 0)
        {
            printf("ERROR: Invalid typeid at offset %s\n", user_query.offset);
            fclose(emplptr);
            fclose(fileptr);
            ret_value = -1;
            return((char *) &ret_value);
        }
        /* If the typeid is not an employee information, then return error */
        else if(typeid != EMPL_W2EMP)
        {
            printf("ERROR: Invalid typeid at offset %s\n", user_query.offset);
            fclose(emplptr);
            fclose(fileptr);
            ret_value = -1;
            return((char *) &ret_value);
        }
        /* read in the employee detail record -- If fails return error */
        else if(fread(&employee_detail, sizeof(struct EMPL_W2EMPL_DETAIL),
            1, emplptr) == 0)
        {
            printf("ERROR: Cannot read detail file\n");
            fclose(emplptr);
            fclose(fileptr);
            ret_value = -1;
            return((char *) &ret_value);
        }
        /* write the employee detail record to the output file
        else if(fwrite(&employee_detail, sizeof(struct EMPL_W2EMPL_DETAIL),
            1, fileptr) == 0)
        {
            printf("ERROR: Cannot write to output file\n");
            fclose(emplptr);
            fclose(fileptr);
            ret_value = -1;
            return((char *) &ret_value);
        }
        /* return success
        fclose(emplptr);
        fclose(fileptr);
    }
}
```

```

# Bottom Level Makefile (~/.ssapilot/src/bin/search_header)
#
# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependencies for the source files.

# This is a list of the key directories in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE = search_header
SRC = search_header.c
LIBS = $(LIBDIR)/libgen_eamdate.a
CLIBS = -lgen_eamdate -lm
OBJ = search_header.o
CFLAGS = -I$(INCDIR) -L$(LIBDIR)
CC = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean : rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
rm -f $(EXECUTABLE)
rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependencies for each of the source files
depend : $(SRC)
sed -e '1,/^# DO NOT DELETE/'d' Makefile > mm.tmp
cat dependlist >> mm.tmp
mv Makefile Makefile.bak
mv mm.tmp Makefile
rm -f dependlist

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
$(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
cp $(EXECUTABLE) $(BINDIR)

.c.o : $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
search_header.o: search_header.c
search_header.o: /usr/include/stdio.h
search_header.o: /usr/include/stdlib.h
search_header.o: /usr/include/sys/types.h
search_header.o: /usr/include/sys/select.h
search_header.o: /usr/include/sys/time.h
search_header.o: /usr/include/sys/types.h
search_header.o: /usr/include/time.h
search_header.o: /usr/include/sys/sysmacros.h
search_header.o: /usr/include/sys/time.h
search_header.o: /usr/include/malloc.h
search_header.o: /usr/include/math.h
search_header.o: /usr/include/float.h
search_header.o: /usr/include/ieeefp.h
search_header.o: /usr/include/fcntl.h
search_header.o: /usr/include/rpc/rpc.h
search_header.o: /usr/include/rpc/types.h
search_header.o: /usr/include/sys/types.h
search_header.o: /usr/include/sys/time.h
search_header.o: /usr/include/tuser.h
search_header.o: /usr/include/sys/tuser.h
search_header.o: /usr/include/memory.h
search_header.o: /usr/include/rpc/xdr.h
search_header.o: /usr/include/sys/byteorder.h
search_header.o: /usr/include/sys/types.h
search_header.o: /usr/include/stdio.h
search_header.o: /usr/include/rpc/auth.h
search_header.o: /usr/include/rpc/xdr.h
search_header.o: /usr/include/sys/cred.h
search_header.o: /usr/include/sys/t_lock.h
search_header.o: /usr/include/sys/machlock.h
search_header.o: /usr/include/sys/types.h
search_header.o: /usr/include/sys/dk_lkinfo.h
search_header.o: /usr/include/sys/types.h
search_header.o: /usr/include/sys/dl.h
search_header.o: /usr/include/sys/sleepq.h
search_header.o: /usr/include/sys/turnstile.h
search_header.o: /usr/include/sys/types.h
search_header.o: /usr/include/sys/param.h
search_header.o: /usr/include/limits.h
search_header.o: /usr/include/unistd.h
search_header.o: /usr/include/sys/fcntl.h
search_header.o: /usr/include/sys/signal.h
search_header.o: /usr/include/vm/faultcode.h
search_header.o: /usr/include/sys/types.h
search_header.o: /usr/include/sys/pfrec.h
search_header.o: /usr/include/sys/sleepq.h
search_header.o: /usr/include/sys/mutex.h
search_header.o: /usr/include/sys/types.h
search_header.o: /usr/include/sys/machlock.h
search_header.o: /usr/include/sys/turnstile.h

```

```
search_header.o: /usr/include/sys/dk_lkinfo.h
search_header.o: /usr/include/rpc/clnt.h
search_header.o: /usr/include/rpc/rpc_com.h
search_header.o: /usr/include/sys/netconfig.h
search_header.o: /usr/include/rpc/rpc_msg.h
search_header.o: /usr/include/rpc/clnt.h
search_header.o: /usr/include/rpc/auth_sys.h
search_header.o: /usr/include/rpc/auth_des.h
search_header.o: /usr/include/rpc/auth_kerb.h
search_header.o: /usr/include/kerberos/krb.h
search_header.o: /usr/include/kerberos/mit-copyright.h
search_header.o: /usr/include/kerberos/des.h
search_header.o: /usr/include/kerberos/mit-copyright.h
search_header.o: /usr/include/sys/socket.h
search_header.o: /usr/include/sys/netconfig.h
search_header.o: /usr/include/netinet/in.h
search_header.o: /usr/include/sys/stream.h
search_header.o: /usr/include/sys/vnode.h
search_header.o: /usr/include/sys/types.h
search_header.o: /usr/include/sys/t_lock.h
search_header.o: /usr/include/sys/time.h
search_header.o: /usr/include/sys/cred.h
search_header.o: /usr/include/sys/uio.h
search_header.o: /usr/include/sys/types.h
search_header.o: /usr/include/sys/poll.h
search_header.o: /usr/include/sys/strndep.h
search_header.o: /usr/include/sys/cred.h
search_header.o: /usr/include/sys/t_lock.h
search_header.o: /usr/include/sys/bt_eorder.h
search_header.o: /usr/include/rpc/svc.h
search_header.o: /usr/include/rpc/rpc_com.h
search_header.o: /usr/include/rpc/rpc_msg.h
search_header.o: /usr/include/rpc/svc_auth.h
search_header.o: /usr/include/rpc/svc.h
search_header.o: /usr/include/rpc/rpcb_clnt.h
search_header.o: /usr/include/rpc/types.h
search_header.o: /usr/include/rpc/rpcb_prot.h
search_header.o: ../../include/params.h
search_header.o: ../../include/eamatestruct.h
search_header.o: ../../include/btreestruct.h
```



```

/*
 * search_header.c
 * version 4.0
 * 10/26/93
 *
 * by Natalie Willman
 *
 * This module contains the main search control routines for searching
 * for employer header information, and its auxiliary functions.
 *
 */

/* Include Files */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/time.h>
#include <malloc.h>
#include <math.h>
#include <fcntl.h>
#include <rpc/rpc.h>
#include "params.h"
#include "eamatestruct.h" /* Data file structure definitions */
#include "btreetstruct.h" /* B+ tree structure definitions */

/*
 * MAIN PROGRAM
 *
 * The main program registers the rpc call(s) and enters into
 * a wait loop for them.
 *
 * Listing of Functions and Prototypes:
 */

char *get_emplr_header();
char *get_seq_header();

main()
{
    int rep;

    /* Register the rpc calls and check for an error return
    rep = register_rpc(0x30000000L, 1L, 1L, get_emplr_header, xdr_int, xdr_int);
    if (rep == -1)
    {
        printf("ERROR: Cannot register search_header\n");
        perror("register_rpc(get_emplr_hdr)");
    }

    rep = register_rpc(0x36000000L, 1L, 1L, get_seq_header, xdr_int, xdr_int);
    if (rep == -1)
    {
        printf("ERROR: Cannot register search_header\n");
        perror("register_rpc(get_seq_header)");
    }

    /* enter a wait loop - waiting for a request from the user
    svc_run();

    /* SHOULD NEVER REACH HERE !!!
    printf("ERROR: returned from svc_run() in search_header\n");
    }

    perror("svc_run()");
    }

    *****
    /* Functions for get_emplr_header()
    /*
    /* This function accepts as input from the user a query structure
    /* containing a year and ein. It searches the master employer header
    /* file for all headers for this year and ein and prints them to an
    /* output file.
    /*
    /*
    char *get_emplr_header(user_num)
    int *user_num;
    {
        int ret_value;
        char namestring[FILENAME];
        struct USER_QUERY user_query;
        FILE *fileptr;
        long browse_offset;

        printf("Employer Header Request -- all\n");

        /* Open the user query file
        sprintf(namestring, "query%d.txt", *user_num);
        fileptr = fopen(namestring, "rb");
        if (fileptr == NULL)
        {
            printf("ERROR: Cannot open file %s\n", namestring);
            ret_value = -1;
            return((char *) &ret_value);
        }

        /* Read the user query
        if (fread(&user_query, sizeof(struct USER_QUERY), 1, fileptr) == 0)
        {
            printf("ERROR: Cannot read query\n");
            fclose(fileptr);
            ret_value = -1;
            return((char *) &ret_value);
        }

        /* print the query parameters -- for debugging purposes
        printf("user: %d\n", user_num);
        printf("Year: %s\n", user_query.year);
        printf("EIN: %s\n", user_query.ein);
        printf("EST: %s\n", user_query.est);
        fclose(fileptr);

        /* open the user header file -- to contain the employer headers
        sprintf(namestring, "hdr%d.txt", *user_num);
        fileptr = fopen(namestring, "w");
        if (fileptr == NULL)
        {
            printf("ERROR: Cannot open file %s\n", namestring);
            ret_value = -1;
            return((char *) &ret_value);
        }

```

```

/* search for the offset to the first employer header for this eln
/* If this returns a -1, there is no such eln in the index file
browse_offset = search_all_emplr(user_query.year, user_query.eln, fileptr);
if(browse_offset != -1)
{
    /* close the files, and return a successful response
    fclose(fileptr);
    ret_value = 1;
    return((char *) &ret_value);
}
else
{
    /* If the eln does not exist, close the files, and return an
    /* unsuccessful response
    /* unsuccessful response
    fclose(fileptr);
    printf("ERROR: Employer header not found for this ELN\n");
    ret_value = -1;
    return((char *) &ret_value);
}
}

/*****
/* Functions for get_seq_header()
/*
/* This function accepts as input from the user a query structure
/* containing a year, eln and sequence number. It searches the master
/* employer header file for the header with this year, eln and seq no
/* and prints it to an output file.
/*
/*
/*****/

char *get_seq_header(user_num)
int *user_num;
{
    int ret_value;
    char namestring[FILENAME];
    struct USER_QUERY user_query;
    FILE *fileptr;
    long browse_offset;

    printf("Employer Header Request\n");

    /* Open the user query structure file
    sprintf(namestring, "query%d.txt", *user_num);
    fileptr = fopen(namestring, "rb");
    if(fileptr == NULL)
    {
        printf("ERROR: Cannot open file %s\n", namestring);
        ret_value = -1;
        return((char *) &ret_value);
    }

    /* Read in the user query structure
    if(fread(&user_query, sizeof(struct USER_QUERY), 1, fileptr) == 0)
    {
        printf("ERROR: Cannot read query\n");
        fclose(fileptr);
        ret_value = -1;
        return((char *) &ret_value);
    }
}

```

```

/* Print the user query parameters -- for debugging information
printf("User: %d\n", *user_num);
printf("Year: %s\n", user_query.year);
printf("EIN: %s\n", user_query.eln);
printf("EST: %s\n", user_query.est);
printf("SEQ_NO: %s\n", user_query.seq_no);
fclose(fileptr);

/* Open the output file, where the header information will be printed
sprintf(namestring, "hdr%d.txt", *user_num);
fileptr = fopen(namestring, "w");
if(fileptr == NULL)
{
    printf("ERROR: Cannot open file %s\n", namestring);
    ret_value = -1;
    return((char *) &ret_value);
}

/* search for the employer header matching the year, eln and sequence
/* number and write it to the output file
browse_offset = search_seq_emplr(user_query.year, user_query.eln,
user_query.seq_no, fileptr);
if(browse_offset != -1)
{
    fclose(fileptr);
    ret_value = 1;
    return((char *) &ret_value);
}
else
{
    fclose(fileptr);
    printf("ERROR: Employer header not found for this sequence\n");
    ret_value = -1;
    return((char *) &ret_value);
}
}

```

```

# Bottom Level Makefile (~/.ssapilot/src/bin/search_print)

# This make file is at the lowest level in the
# project hierarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependencies for the source files.

# This is a list of the key directories in the
# project hierarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../..
LIBDIR = $(PROJECT_ROOT)/lib
INCDIR = $(PROJECT_ROOT)/include
BINDIR = $(PROJECT_ROOT)/bin

# This is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE = search_print
SRC = search_print.c
LIBS = $(LIBDIR)/libgen.eamdate.a
CLIBS = -lgen.eamdate -lm
OBJ = search_print.o
CFLAGS = -I$(INCDIR) -L$(LIBDIR)
CC = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependencies for each of the source files
depend : $(SRC)
    sed -e '1,/^# DO NOT DELETE/Id' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv Makefile Makefile.bak
    mv mm.tmp Makefile
    rm -f dependlist

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
    $(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(BINDIR)

.c.o :
    $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
search_print.o: search_print.c
search_print.o: /usr/ucbinclude/stdio.h
search_print.o: /usr/include/stdlib.h
search_print.o: /usr/ucbinclude/sys/types.h
search_print.o: /usr/include/sys/select.h
search_print.o: /usr/include/sys/time.h
search_print.o: /usr/ucbinclude/sys/types.h
search_print.o: /usr/include/time.h
search_print.o: /usr/ucbinclude/sys/sysmacros.h
search_print.o: /usr/include/sys/time.h
search_print.o: /usr/include/malloc.h
search_print.o: /usr/include/math.h
search_print.o: /usr/include/floatngpoint.h
search_print.o: /usr/include/sys/leefp.h
search_print.o: /usr/ucbinclude/fcntl.h
search_print.o: /usr/include/rpc/rpc.h
search_print.o: /usr/include/rpc/types.h
search_print.o: /usr/ucbinclude/rpc/types.h
search_print.o: /usr/ucbinclude/sys/time.h
search_print.o: /usr/include/tuser.h
search_print.o: /usr/include/sys/tuser.h
search_print.o: /usr/ucbinclude/fcntl.h
search_print.o: /usr/include/memory.h
search_print.o: /usr/include/rpc/xdr.h
search_print.o: /usr/include/sys/byteorder.h
search_print.o: /usr/ucbinclude/sys/types.h
search_print.o: /usr/ucbinclude/stdio.h
search_print.o: /usr/include/rpc/auth.h
search_print.o: /usr/include/rpc/xdr.h
search_print.o: /usr/include/sys/cred.h
search_print.o: /usr/include/sys/t_lock.h
search_print.o: /usr/include/sys/machlock.h
search_print.o: /usr/ucbinclude/sys/types.h
search_print.o: /usr/ucbinclude/dk_lkinfo.h
search_print.o: /usr/ucbinclude/sys/types.h
search_print.o: /usr/include/sys/dl.h
search_print.o: /usr/ucbinclude/unistd.h
search_print.o: /usr/ucbinclude/sys/fcntl.h
search_print.o: /usr/ucbinclude/sys/signal.h
search_print.o: /usr/include/vm/faultcode.h
search_print.o: /usr/ucbinclude/sys/types.h
search_print.o: /usr/include/sys/piprec.h
search_print.o: /usr/include/sys/sleepq.h
search_print.o: /usr/include/sys/mutex.h
search_print.o: /usr/ucbinclude/sys/types.h
search_print.o: /usr/include/sys/machlock.h
search_print.o: /usr/include/sys/turnstile.h

```



```
search_print.o: /usr/include/sys/dk1_lkinfo.h
search_print.o: /usr/include/rpc/clnt.h
search_print.o: /usr/include/rpc/rpc_com.h
search_print.o: /usr/include/sys/netconfig.h
search_print.o: /usr/include/rpc/rpc_msg.h
search_print.o: /usr/include/rpc/clnt.h
search_print.o: /usr/include/rpc/auth_sys.h
search_print.o: /usr/include/rpc/auth_des.h
search_print.o: /usr/include/rpc/auth_kerb.h
search_print.o: /usr/include/kerberos/krb.h
search_print.o: /usr/include/kerberos/mit-copyright.h
search_print.o: /usr/include/kerberos/des.h
search_print.o: /usr/include/kerberos/mit-copyright.h
search_print.o: /usr/include/sys/socket.h
search_print.o: /usr/include/sys/netconfig.h
search_print.o: /usr/include/netinet/in.h
search_print.o: /usr/include/sys/stream.h
search_print.o: /usr/include/sys/vnode.h
search_print.o: /usr/include/sys/types.h
search_print.o: /usr/include/sys/t_lock.h
search_print.o: /usr/include/sys/time.h
search_print.o: /usr/include/sys/cred.h
search_print.o: /usr/include/sys/uio.h
search_print.o: /usr/include/sys/types.h
search_print.o: /usr/include/sys/poll.h
search_print.o: /usr/include/sys/strmdp.h
search_print.o: /usr/include/sys/cred.h
search_print.o: /usr/include/sys/t_lock.h
search_print.o: /usr/include/sys/byteorder.h
search_print.o: /usr/include/rpc/svc.h
search_print.o: /usr/include/rpc/rpc_com.h
search_print.o: /usr/include/rpc/rpc_msg.h
search_print.o: /usr/include/rpc/svc_auth.h
search_print.o: /usr/include/rpc/svc.h
search_print.o: /usr/include/rpc/rpcb_clnt.h
search_print.o: /usr/include/rpc/types.h
search_print.o: /usr/include/rpc/rpcb_prot.h
search_print.o: ../..//include/params.h
search_print.o: ../..//include/eamestruct.h
```

```

/* search_print.c
 * version 4.0
 * 10/26/93
 * by Natalie Willman
 *
 * This module contains the main print control routine, and its
 * auxiliary functions.
 *
 * Functions for print detail were written by Ken Davidson of SSA,
 * and were modified by Natalie Willman to: Incorporate into the
 * network format, to remove global variables, and to make other
 * minor changes.
 */

/* Include Files */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/time.h>
#include <malloc.h>
#include <math.h>
#include <fcntl.h>
#include <rpc/rpc.h>
#include "params.h"
#include "eamat:struct.h" /* Data file structure definitions */

/*
 * MAIN PROGRAM
 *
 * This program will register the rpc call and enter a wait loop
 * for requests.
 *
 * Listing of Functions and Prototypes:
 */
char *print_report();

main()
{
    int rep;

    /* register the rpc call and check for an error return
     rep = register_rpc(0x33000000L, 1L, 1L, print_report, xdr_int, xdr_int);
     if (rep == -1)
     {
         printf("ERROR: Cannot register search_print\n");
         perror("register_rpc(print_report)");
     }

     /* enter a wait loop - wait for a request from the user
     svc_run();

     /* SHOULD NEVER REACH HERE
     printf("ERROR: returned from svc_run() in search_print\n");
     perror("svc_run()");
     }

```

```

{
    /* read in the employer information and store it in a structure */
    fseek(outptr, 0, 0);
    read_eamate_w2header_info(outptr, &employer_info);
    fclose(outptr);

    /* create the filename for an open the employer detail file */
    if (employer_info.num_recs < 5000)
    {
        cr_detail_filename(namestring, user_query.year, user_query.eln,
            user_query.seq_no, employer_info.platter_side);

        /* open the output file, and print the detail file to it (formatted) */
        sprintf(printfile, "print%d.txt", "user_num");
        ret_value = print_detail(namestring, printfile);
        sprintf(commandstring, "lp %s", printfile);
        system(commandstring);
        return((char *) &ret_value);
    }
    else
    {
        printf("ERROR: Employer Report Too Large to Print -- See System Administrator\n");
        ret_value = -1;
        return((char *) &ret_value);
    }
}
else
{
    /* If the report did not exist, then print an message and return error */
    fclose(outptr);
    printf("ERROR: Employer header not found for this sequence\n");
    ret_value = -1;
    return((char *) &ret_value);
}
}

/*
 * print_detail()
 * This function will take an input file (employer detail file)
 * and an output file, and will format the detail file as it was
 * on microfilm and print the file
 *
 * Input: Input file name
 * Output file name
 */

print_detail(inputfile, outputfile)
char inputfile[], outputfile[];
{
    FILE *data;
    FILE *testfile;
    char typeid;
    int whole = 0;
    int rem = 0;
    static int recpp = 11;
    int index0 = 0;
    int index1 = 0;
    int index2 = 0;
    int index3 = 0;
}

```

```

struct EAMATE_W2EMPLR_DETAIL empl;
struct EAMATE_W2EMPLR_HEADER emplr;
struct EAMATE_W2INTERMED_TOT inter;
struct EAMATE_W2FINAL_TOT final;

testfile = fopen(outputfile, "w");
if (testfile == NULL)
    return(-1);

/* open the file of records to read
data = fopen(inputfile, "r");
if (data == NULL)
{
    printf("ERROR: Cannot open file %s\n", inputfile);
    return(-1);
}

/* While there are still records, read in the record id, and */
/* based upon it, read in the appropriate record and print it */
/* as well as the offset to the record in the data file */
while (!feof(data))
{
    if (fread(&typeid, sizeof(char), 1, data) != 0)
        switch (typeid)
        {
            case MATE_W2ET: /* Employee Information record */
                if (index1 != 0)
                {
                    whole = index1 / recpp;
                    rem = index1 - (recpp * whole);
                    if (rem == 0)
                        print_employer_rec(emplr, testfile);
                }
                fread(&empl, sizeof(struct EAMATE_W2EMPLR_DETAIL), 1, data);
                print_employee_rec(empl, testfile);
                index1++;
                break;

            case MATE_W2EH: /* Employer Header record */
                fread(&emplr, sizeof(struct EAMATE_W2EMPLR_HEADER), 1, data);
                print_employer_rec(emplr, testfile);
                index0++;
                break;

            case MATE_W2IT: /* Intermediate total record */
                fread(&inter, sizeof(struct EAMATE_W2INTERMED_TOT), 1, data);
                print_intermed_rec(inter, testfile);
                index2++;
                break;

            case MATE_W2FT: /* Final total Record */
                fread(&final, sizeof(struct EAMATE_W2FINAL_TOT), 1, data);
                print_final_rec(final, testfile);
                index3++;
                break;
        }
    }
    fclose(testfile);
    return(1);
}

```



```

/*
 * print_employee_rec()
 *
 * *****
 * This record prints an employee information record on the
 * screen
 *
 * Input: structure containing the employee record information
 *
 * Output: The employee record is printed
 *
 * *****
 */
print_employee_rec(empl_rec, testfile)
struct EAMATE_W2EMPLR_DETAIL empl_rec;
FILE *testfile;
{
    fprintf(testfile, "%s", empl_rec.ssn); /* Soc. Sec. No. */
    fprintf(testfile, "%s", empl_rec.name); /* Employee Name */
    fprintf(testfile, "%s", empl_rec.pens_ind); /* Pension Indicator */
    fprintf(testfile, "%s", empl_rec.defcomp_ind); /* Def Comp Ind */
    fprintf(testfile, "%s", empl_rec.wages); /* Ann. FICA Wages */
    fprintf(testfile, "%s", empl_rec.tips); /* Ann. FICA Tips */
    fprintf(testfile, "%s", empl_rec.other); /* Other Compensation */
    fprintf(testfile, "%s", empl_rec.fedtax); /* Federal Tax */
    fprintf(testfile, "%s", empl_rec.fica_tax); /* FICA Tax */
    fprintf(testfile, "%s", empl_rec.adv_earn_inc); /* Adv. Earned Income */
    fprintf(testfile, "%s", empl_rec.med_wages); /* Medicare Wages */
    fprintf(testfile, "%s", empl_rec.med_tax); /* Medicare Tax */
    fprintf(testfile, "%s", empl_rec.ctrl_no); /* Control Number */
    fprintf(testfile, "%s", empl_rec.street_add); /* Str Address */
    fprintf(testfile, "%s", empl_rec.dep_care); /* Dependent Care */
    fprintf(testfile, "%s", empl_rec.alloc_tips); /* Allocation Tips */
    fprintf(testfile, "%s", empl_rec.insur); /* GRP-TERM-INS-%s, empl_rec.grp_insur; /* Grp Insur */
    fprintf(testfile, "%s", empl_rec.uncoll_fica_tax); /* Uncoll FICA Tax */
    fprintf(testfile, "%s", empl_rec.city); /* City */
    fprintf(testfile, "%s", empl_rec.state); /* State */
    fprintf(testfile, "%s", empl_rec.zip_code); /* Zip Code */
    fprintf(testfile, "%s", empl_rec.def_comp); /* Def Comp */
    fprintf(testfile, "%s", empl_rec.stal); /* Station */
    fprintf(testfile, "%s", empl_rec.fr_ben); /* Fr. Ben. */
    fprintf(testfile, "%s", empl_rec.spacing); /* spacing */
    fprintf(testfile, "%s", empl_rec.nqsec); /* NQSEC457= */
    fprintf(testfile, "%s", empl_rec.nqnot); /* NQNOT */
}

/*
 * print_employee_rec()
 *
 * *****
 * This record prints an employer header record on the
 * screen
 *
 * Input: structure containing the employer header record
 *
 * Output: The employer record is printed
 *
 * *****
 */
print_employee_rec(empl_rec, testfile)
struct EAMATE_W2EMPLR_HEADER empl_rec;
FILE *testfile;
{
    fprintf(testfile, "%f", empl_rec.ein); /* Go to a new page */
    fprintf(testfile, "%s", empl_rec.est); /* EST */
    fprintf(testfile, "%s", empl_rec.rpt_yr); /* Year of Report */
    fprintf(testfile, "%s", empl_rec.proc_yr); /* Yr Processed */
    fprintf(testfile, "%s", empl_rec.lib_num); /* TL */
    fprintf(testfile, "%s", empl_rec.type_emplr); /* Type of Emplr */
    fprintf(testfile, "%s", empl_rec.name_code); /* Name Code */
    fprintf(testfile, "%s", empl_rec.other_ein); /* Other EIN */
    fprintf(testfile, "%s", empl_rec.mrn); /* Record Number */
    fprintf(testfile, "%s", empl_rec.name); /* Employer Name */
    fprintf(testfile, "%s", empl_rec.street_add); /* Street Address */
    fprintf(testfile, "%s", empl_rec.city); /* City */
    fprintf(testfile, "%s", empl_rec.state); /* State */
    fprintf(testfile, "%s", empl_rec.zip_code); /* Zip Code */
    fprintf(testfile, "%s", empl_rec.annual_advance); /* ANNUAL ADVANCE */
    fprintf(testfile, "%s", empl_rec.fica_wages_tips); /* FICA WAGES/TIPS */
    fprintf(testfile, "%s", empl_rec.federal_fica_earned_wages_medicare); /* FEDERAL FICA EARNED WAGES MEDICARE */
    fprintf(testfile, "%s", empl_rec.security_no_name_and_address); /* SECURITY NO NAME AND ADDRESS */
    fprintf(testfile, "%s", empl_rec.wages_tips_other_comp); /* WAGES TIPS OTHER-COMP */
    fprintf(testfile, "%s", empl_rec.tax_w_h_income_and_tips_tax); /* TAX W/H INCOME AND TIPS TAX */
    fprintf(testfile, "%s", empl_rec.number); /* NUMBER */
}

/*
 * print_intermed_rec()
 *
 * *****
 * This record prints an intermediate total record on the
 * screen
 *
 * Input: structure containing the intermediate total record
 *
 * Output: The intermediate total record is printed
 *
 * *****
 */
print_intermed_rec(empl_rec, testfile)
struct EAMATE_W2INTERMED_TOT empl_rec;
FILE *testfile;
{
    fprintf(testfile, "%s", empl_rec.totals); /* TOTALS */
    fprintf(testfile, "%s", empl_rec.fica_wages_fica_tips_wg_tp_other); /* FICA WAGES FICA TIPS WG/TP/OTHER */
    fprintf(testfile, "%s", empl_rec.fed_tax_w_h_fica_tax_w_h_earned_inccr); /* FED TAX W/H FICA TAX W/H EARNED INCCR */
    fprintf(testfile, "%s", empl_rec.def_comp_nonqual_457_control); /* DEF COMP NONQUAL 457 CONTROL */
    fprintf(testfile, "%s", empl_rec.intermediate); /* INTERMEDIATE */
}

```

```

fprintf(testfile, " PROCESSED ");
fprintf(testfile, "%s", empl_rec.proc_wages); /* proc FICA WAGES */
fprintf(testfile, "%s", empl_rec.proc_tips); /* proc FICA TIPS */
fprintf(testfile, "%s", empl_rec.proc_other); /* p OTHER WAGES/TIPS */
fprintf(testfile, "%s", empl_rec.proc_fed_tax); /* p FED TAX W/H */
fprintf(testfile, "%s", empl_rec.proc_fica_tax); /* proc FICA TAX WH */
fprintf(testfile, "%s", empl_rec.proc_earn_inc); /* p EARNED INCOME */
fprintf(testfile, "%s", empl_rec.proc_defcomp); /* p DEF COMP */
fprintf(testfile, "%s", empl_rec.proc_nonqual); /* p NONQUAL 457 */

fprintf(testfile, " REPORTED ");
fprintf(testfile, "%s", empl_rec.rep_wages); /* rep FICA WAGES */
fprintf(testfile, "%s", empl_rec.rep_tips); /* rep FICA TIPS */
fprintf(testfile, "%s", empl_rec.rep_other); /* r OTHER WAGES/TIPS */
fprintf(testfile, "%s", empl_rec.rep_fed_tax); /* rep FED TAX W/H */
fprintf(testfile, "%s", empl_rec.rep_fica_tax); /* rep FICA TAX WH */
fprintf(testfile, "%s", empl_rec.rep_earn_inc); /* r EARNED INCOME */
fprintf(testfile, "%s", empl_rec.rep_defcomp); /* r DEF COMP */
fprintf(testfile, "%s", empl_rec.rep_nonqual); /* r NONQUAL 457 */

fprintf(testfile, " TOTALS ");
fprintf(testfile, "MEDIC WAGES MEDIC TAX\n");
fprintf(testfile, " INTERMEDIATE\n");
fprintf(testfile, " PROCESSED ");
fprintf(testfile, "%s", empl_rec.proc_med_wages); /* p MEDIC WAGES */
fprintf(testfile, "%s", empl_rec.proc_med_tax); /* p MEDIC TAX */

fprintf(testfile, " REPORTED ");
fprintf(testfile, "%s", empl_rec.rep_med_wages); /* r MEDIC WAGES */
fprintf(testfile, "%s", empl_rec.rep_med_tax); /* r MEDIC TAX */
}

/* print final rec()
*****
* This record prints a final total record
*
* Input: structure containing the final total record
*
* Output: The final total record is printed
*****
*/

print_final_rec(empl_rec, testfile)
struct EAMATE_W2FINAL_TOT empl_rec;
FILE *testfile;
{
    fprintf(testfile, "TOTALS ");
    fprintf(testfile, "FICA WAGES FICA TIPS WAGES/TIPS/OTHER ");
    fprintf(testfile, "TAX W/H FICA TAX W/H ADVANCED EIC ITEMS\n");
    fprintf(testfile, " EMPLOYER\n");
    fprintf(testfile, " PROCESSED ");
    fprintf(testfile, "%s", empl_rec.proc_wages); /* proc FICA WAGES */
    fprintf(testfile, "%s", empl_rec.proc_tips); /* proc FICA TIPS */
    fprintf(testfile, "%s", empl_rec.proc_other); /* p OTHER WAGES/TIPS */
    fprintf(testfile, "%s", empl_rec.proc_fed_tax); /* p FED TAX W/H */
    fprintf(testfile, "%s", empl_rec.proc_fica_tax); /* proc FICA TAX WH */
}

```

```

fprintf(testfile, "%s", empl_rec.proc_earn_inc); /* p EARNED INCOME */
fprintf(testfile, "%s", empl_rec.proc_items); /* proc ITEMS */

fprintf(testfile, " REPORTED ");
fprintf(testfile, "%s", empl_rec.rep_wages); /* rep FICA WAGES */
fprintf(testfile, "%s", empl_rec.rep_tips); /* rep FICA TIPS */
fprintf(testfile, "%s", empl_rec.rep_other); /* r OTHER WAGES/TIPS */
fprintf(testfile, "%s", empl_rec.rep_fed_tax); /* r FED TAX W/H */
fprintf(testfile, "%s", empl_rec.rep_fica_tax); /* r FICA TAX WH */
fprintf(testfile, "%s", empl_rec.rep_earn_inc); /* r EARNED INCOME */
fprintf(testfile, "%s", empl_rec.rep_items); /* r ITEMS */

fprintf(testfile, "TOTALS ");
fprintf(testfile, "DEF COMP NONQUAL 457 ");
fprintf(testfile, "MEDICARE WAGES MEDICARE TAXES\n");
fprintf(testfile, " EMPLOYER\n");
fprintf(testfile, " PROCESSED ");
fprintf(testfile, "%s", empl_rec.proc_defcomp); /* proc DEF COMP */
fprintf(testfile, "%s", empl_rec.proc_nonqual); /* proc NONQUAL 457 */
fprintf(testfile, "%s", empl_rec.proc_med_wages); /* proc MEDIC WAGES */
fprintf(testfile, "%s", empl_rec.proc_med_tax); /* proc MEDIC TAX */

fprintf(testfile, " REPORTED ");
fprintf(testfile, "%s", empl_rec.rep_defcomp); /* rep DEF COMP */
fprintf(testfile, "%s", empl_rec.rep_nonqual); /* rep NONQUAL 457 */
fprintf(testfile, "%s", empl_rec.rep_med_wages); /* rep MEDIC WAGES */
fprintf(testfile, "%s", empl_rec.rep_med_tax); /* rep MEDIC TAX */
}

```

```

# Bottom Level Makefile (-/ssapilot/src/bin/search_single)
#
# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependancies for the source files.
#
# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../../..
LIBDIR      = $(PROJECT_ROOT)/lib
INCDIR      = $(PROJECT_ROOT)/include
BINDIR      = $(PROJECT_ROOT)/bin

# this is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE  = search_single
SRC          = search_single.c
LIBS        = $(LIBDIR)/libgen, eamata.a $(LIBDIR)/libtree, data.a
CLIBS       = -lgen, eamata -lm -lbtree, data
OBJ         = search_single.o
CFLAGS      = -I$(INCDIR) -I$(LIBDIR)
CC          = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependancies for each of the source files
depend : $(SRC)
    $(CC) -M $(CFLAGS) $(SRC) > dependlist
    sed -e '1,/^# DO NOT DELETE/!d' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv Makefile Makefile.bak
    mv mm.tmp Makefile
    rm -f dependlist

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
    $(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(BINDIR)

.c.o :
    $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
search_single.o: search_single.c
search_single.o: /usr/include/stdio.h
search_single.o: /usr/include/stdlib.h
search_single.o: /usr/include/sys/types.h
search_single.o: /usr/include/sys/select.h
search_single.o: /usr/include/sys/time.h
search_single.o: /usr/include/sys/types.h
search_single.o: /usr/include/time.h
search_single.o: /usr/include/sys/sysmacros.h
search_single.o: /usr/include/sys/time.h
search_single.o: /usr/include/malloc.h
search_single.o: /usr/include/math.h
search_single.o: /usr/include/floatnpoint.h
search_single.o: /usr/include/sys/ieee.h
search_single.o: /usr/include/fcntl.h
search_single.o: /usr/include/rpc/rpc.h
search_single.o: /usr/include/rpc/types.h
search_single.o: /usr/include/rpc/types.h
search_single.o: /usr/include/sys/time.h
search_single.o: /usr/include/tuser.h
search_single.o: /usr/include/sys/tuser.h
search_single.o: /usr/include/fcntl.h
search_single.o: /usr/include/memory.h
search_single.o: /usr/include/rpc/xdr.h
search_single.o: /usr/include/sys/byteorder.h
search_single.o: /usr/include/sys/types.h
search_single.o: /usr/include/stdio.h
search_single.o: /usr/include/rpc/auth.h
search_single.o: /usr/include/rpc/xdr.h
search_single.o: /usr/include/sys/cred.h
search_single.o: /usr/include/sys/t_lock.h
search_single.o: /usr/include/sys/machlock.h
search_single.o: /usr/include/sys/types.h
search_single.o: /usr/include/sys/dk1_kinfo.h
search_single.o: /usr/include/sys/types.h
search_single.o: /usr/include/sys/di.h
search_single.o: /usr/include/sys/sleepq.h
search_single.o: /usr/include/sys/turnstile.h
search_single.o: /usr/include/sys/types.h
search_single.o: /usr/include/sys/param.h
search_single.o: /usr/include/limits.h
search_single.o: /usr/include/unistd.h
search_single.o: /usr/include/sys/fcntl.h
search_single.o: /usr/include/sys/signal.h
search_single.o: /usr/include/vm/faultcode.h
search_single.o: /usr/include/sys/types.h
search_single.o: /usr/include/sys/pirec.h
search_single.o: /usr/include/sys/sleepq.h
search_single.o: /usr/include/sys/mutex.h
search_single.o: /usr/include/sys/types.h
search_single.o: /usr/include/sys/machlock.h
search_single.o: /usr/include/sys/turnstile.h

```



```
search_single.o: /usr/include/sys/dkinfo.h
search_single.o: /usr/include/rpc/clnt.h
search_single.o: /usr/include/rpc/rpc_com.h
search_single.o: /usr/include/sys/netconfig.h
search_single.o: /usr/include/rpc/rpc_msg.h
search_single.o: /usr/include/rpc/clnt.h
search_single.o: /usr/include/rpc/auth_sys.h
search_single.o: /usr/include/rpc/auth_des.h
search_single.o: /usr/include/rpc/auth_kerb.h
search_single.o: /usr/include/kerberos/krb.h
search_single.o: /usr/include/kerberos/mit-copyright.h
search_single.o: /usr/include/kerberos/des.h
search_single.o: /usr/include/kerberos/mit-copyright.h
search_single.o: /usr/include/sys/socket.h
search_single.o: /usr/include/sys/netconfig.h
search_single.o: /usr/include/netinet/in.h
search_single.o: /usr/include/sys/stream.h
search_single.o: /usr/include/sys/vnode.h
search_single.o: /usr/include/sys/types.h
search_single.o: /usr/include/sys/t_lock.h
search_single.o: /usr/include/sys/time.h
search_single.o: /usr/include/sys/cred.h
search_single.o: /usr/include/sys/uio.h
search_single.o: /usr/include/sys/types.h
search_single.o: /usr/include/sys/poll.h
search_single.o: /usr/include/sys/rmdep.h
search_single.o: /usr/include/sys/cred.h
search_single.o: /usr/include/sys/t_lock.h
search_single.o: /usr/include/sys/bt_eorder.h
search_single.o: /usr/include/rpc/svc.h
search_single.o: /usr/include/rpc/rpc_com.h
search_single.o: /usr/include/rpc/rpc_msg.h
search_single.o: /usr/include/rpc/svc_auth.h
search_single.o: /usr/include/rpc/svc.h
search_single.o: /usr/include/rpc/rpcb_clnt.h
search_single.o: /usr/include/rpc/types.h
search_single.o: /usr/include/rpc/rpcb_prot.h
search_single.o: /usr/include/rpc/rpcb.h
search_single.o: ../../include/params.h
search_single.o: ../../include/eamatstruct.h
search_single.o: ../../include/btreestruct.h
search_single.o: /usr/include/param.h
```

```

/*
 * search_single.c
 * version 4.0
 * 10/26/93
 *
 * by Natalie Willman
 *
 * This module contains the main search control routine for a
 * single query request, and its auxiliary functions.
 */

/* Include files */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/time.h>
#include <malloc.h>
#include <math.h>
#include <fcntl.h>
#include <rpc/rpc.h>
#include "params.h"
#include "easstruct.h"
#include "btreetstruct.h"
#include "prof.h"

/* Data file structure definitions */
/* B+ tree structure definitions */

/* Main program will register the rpc call and enter a wait
 * loop waiting for requests.
 *
 * Listing of Functions and Prototypes:
 */
char *single_query();

main()
{
    int rep;

    /* register the RPC Call and check for an error return
     rep = register_rpc(0x31000000L, 1L, 1L, single_query, xdr_int, xdr_int);
     if (rep == -1)
     {
         printf("ERROR: Cannot register search_single\n");
         perror("register_rpc(single_query)");
     }

    /* Enter a wait loop -- waiting for a request from the client
     svc_run();

    /* SHOULD NEVER REACH THIS POINT !!!
     printf("ERROR: returned from svc_run() in search_single\n");
     perror("svc_run()");
     }

    *****
    */
    /* Functions for single_query()
    */
}

/* This function accepts as input from the user a query structure
 * containing a year, ein and name/ssn info. It searches the employer
 * report for matches to this information, and ranks and prints the
 * best matches to an output file.
 */
/*****
char *single_query(user_num)
int *user_num;
{
    int ret_value, result; /* result returned from search, and returned to user */
    char namestring[FILENAME]; /* string to hold filenames to be opened */
    char firstinitial[3]; /* array to hold the first initial from first name */
    char filename[FILENAME], /* string to hold a file name */
        name[FILENAME], /* string to hold a file name */
        statname[FILENAME]; /* string to hold a file name */
    struct USER_QUERY user_query; /* user query structure from client request */
    FILE *fileptr, *listptr, *bwptr; /* file pointers for query, list offsets
    FILE *statsptr, *repptr; /* browse data, stats, and detail
    struct TEST_DATA test_info; /* structure to hold the test data
    long browse_offset, timer, timer1, statoffset; /* offset and timer vars
    long *sortloc[DICE_GRAIN]; /* arrays for sorting the offsets and index
    int sortindex[DICE_GRAIN]; /* into the sortloc array
    int i, j, num; /* counter variables
    time_t etimerstart, etimerend, junk; /* timer variables and trash var for timer
    FILE *rankptr;

    printf("Single Query request\n");

    /* Open the user query structure file
    sprintf(namestring, "query%d.txt", *user_num);
    fileptr = fopen(namestring, "rb");
    if (fileptr == NULL)
    {
        printf("ERROR: Cannot open file %s\n", namestring);
        ret_value = -1;
        return((char *) &ret_value);
    }

    /* Open the search statistics file
    sprintf(statname, "search%d.txt", *user_num);
    statsptr = fopen(statname, "a");
    if (statsptr == NULL)
    {
        fclose(fileptr);
        printf("ERROR: Cannot open file %s\n", statname);
        ret_value = -1;
        return((char *) &ret_value);
    }

    /* Read in the user query structure
    if (fread(&user_query, sizeof(struct USER_QUERY), 1, fileptr) == 0)
    {
        printf("ERROR: Cannot read query\n");
        fclose(fileptr);
        fclose(statsptr);
        ret_value = -1;
        return((char *) &ret_value);
    }
    fclose(fileptr);

    /* Copy Relevant Information from the user query structure to the test
    /* Info structure saved for each search

```

```

test_info.user = user_num;
strcpy(test_info.year, user_query.year);
strcpy(test_info.ein, user_query.ein);
strcpy(test_info.first, user_query.first);
strcpy(test_info.last, user_query.last);
strcpy(test_info.ssn, user_query.ssn);
test_info.mem = 0;

/* Print the user query structure -- for debugging information */
printf("User: %d\n", user_num);
printf("Year: %s\n", user_query.year);
printf("Ein: %s\n", user_query.ein);
printf("EST: %s\n", user_query.est);
printf("First: %s\n", user_query.first);
printf("Last: %s\n", user_query.last);
printf("SSN: %s\n", user_query.ssn);

/* Open the output file to write the search results (offset list) to */
sprintf(filename, "listoff%d.txt", user_num);
listptr = fopen(filename, "w+");
if(listptr == NULL)
{
    printf("ERROR: Cannot open file %s\n", filename);
    fclose(statsptr);
    ret_value = -1;
    return((char *) &ret_value);
}

/* NOT IN USE ***** */
/* Open the output file to write the search results (offset list) to */
/* NOT IN USE ***** */
sprintf(filename, "rank%d.txt", user_num);
rankptr = fopen(filename, "w+");
if(rankptr == NULL)
{
    printf("ERROR: Cannot open file %s\n", filename);
    fclose(statsptr);
    ret_value = -1;
    return((char *) &ret_value);
}

/* Open the output file to write the search results (browse data) to */
sprintf(filename, "brow%d.txt", user_num);
brwptr = fopen(filename, "w+");
if(brwptr == NULL)
{
    printf("ERROR: Cannot open file %s\n", filename);
    fclose(statsptr);
    fclose(listptr);
    ret_value = -1;
    return((char *) &ret_value);
}

/* open the report browse file
or browse filename(filename, user_query.year, "0", user_query.ein);
reptr = fopen(filename, "r");
if(reptr == NULL)
{
    printf("ERROR: Cannot open file %s\n", filename);
    fclose(statsptr);
    fclose(listptr);
    fclose(brwptr);
}

test_info.user = user_num;
return((char *) &ret_value);
}

/* Open the employer header file, where header info will be printed */
sprintf(filename, "hdr%d.txt", user_num);
fileptr = fopen(filename, "w");
if(fileptr == NULL)
{
    printf("ERROR: Cannot open file %s\n", filename);
    fclose(statsptr);
    fclose(listptr);
    fclose(brwptr);
    fclose(reptr);
    ret_value = -1;
    return((char *) &ret_value);
}

/* search for record matching criteria entered by the user */
browse_offset = search_seq_empl(user_query.year, user_query.ein, "AAA", fileptr);
fclose(fileptr);
if(browse_offset != -1)
{
    /* create the base filename for the index files */
    create_filename(filename, user_query.year, user_query.ein);

    /* create the search name by adding an initial field from first name */
    firstinitial[0] = ' ';
    firstinitial[1] = user_query.first[0];
    firstinitial[2] = 0;
    strcpy(name, user_query.first);
    strcat(name, firstinitial);
    strcat(name, " ");
    strcat(name, user_query.last);
    strcpy(test_info.searchname, name);

    /* Initialize timers and start the search process, and end timers */
    etimerst = time(&junk);
    timer = clock();
    result = search_record(name, user_query.ssn, filename, sortloc, sortIndex, &test_info, &reptr);
    timer1 = clock();
    etimerend = time(&junk);

    /* store stats info in the test structure, and if there was an error */
    /* write the stats structure to the disk */
    test_info.nummatch = result;
    test_info.ctime = (double) (timer1 - timer) / (double) 1000000;
    test_info.etime = etimerend - etimerst;
    if(result == -1)
    {
        printf("ERROR: Search_Record function returned FAIL\n");
        fwrite(&test_info, sizeof(struct TEST_DATA), 1, statsptr);
        fclose(statsptr);
        fclose(brwptr);
        fclose(reptr);
        fclose(listptr);
        ret_value = -1;
        return((char *) &ret_value);
    }

    /* for each of the queues, write the matches to disk, keeping track */
    /* of the write time needed */
    etimerst = time(&junk);

```



```

for(i = DICE_GRAIN-1; i >= 0; i--)
{
    test_info.mem = test_info.mem + (sizeof(long) * sizeof(long));
    fwrite(sortloc[i], sizeof(long), sortindex[i], listptr);
    /* At this time do not write out the rank information */
    for(j = 0; j < sortindex[i]; j++)
        fwrite(&i, sizeof(long), 1, rankptr);
    free(sortloc[i]);
}

/* translate the first set of offsets into browse data for the client */
ret_value = translate_offsets(listptr, brwptr, reptr, 0);

etimerend = time(&junk);
/* close files, end timers, and write the stats info to disk */
fclose(listptr);
fclose(rankptr);
fclose(brwptr);
fclose(reptr);
test_info.wtime = etimerend - etimerst;
fwrite(test_info, sizeof(struct TEST_DATA), 1, statsptr);
fclose(statsptr);
return((char *) &ret_value);
}
else /* if the report does not exist, return fail */
{
    fclose(statsptr);
    fclose(brwptr);
    fclose(listptr);
    /* fclose(rankptr); */
    fclose(reptr);
    ret_value = -1;
    return((char *) &ret_value);
}
}

/* search_record()
* this function takes the user input, and divides it into grams,
* searches for each of the grams, and organizes the list of
* offsets returned in priority order. The records at these offsets
* are then printed
*/

search_record(namekey, ssnkey, filename, sortloc, sortindex, test_info, reptr)
char filename[], namekey[], ssnkey[];
long *sortloc;
int sortindex[];
struct TEST_DATA *test_info;
FILE *reptr;
{
    long dupe_loc; /* offset of current location in duplicate file */
    int count, i, /* counter variables */
        remaining, /* number of queues returned from the gram search */
        numdups; /* total number of offsets returned from search */
    FILE *indxptr, *ssnptr, *ssndup, *namedup; /* file ptrs */
    char parse_array[ARRAY_SIZE][KEYLEN]; /* array to hold ngrams
    struct RECINFO *locations[ARRAY_SIZE+3]; /* array for queues of locs */
    char trash[COMMANDELENGTH]; /* filename array & concat array */

    /* for each gram, ...
    remaining = count;
    for(i = 0; i < count; i++)
    {
        /* search the b+tree and get an offset to duplicate queue
        dupe_loc = search_tree(parse_array[i], indxptr, &frequency[i]);
        /* pruning based upon the frequency of grams within the file
        if((FPRUNE == TRUE) && (remaining > MIN_GRAMS) &&
            (((double)frequency[i]/(double)numdups)) > FPRUNE_LEVEL))
        {
            dupe_loc = -1;
            remaining--;
            test_info->grprune++;
        }

        /* parse the name key, and return the number of ngrams in the name */
        count = parse_name(namekey, parse_array);
        if(count == 0)
        else
            name_entered = FALSE;
            name_entered = TRUE;

        fseek(reptr, 0, 2);
        numdups = ftell(reptr)/sizeof(struct EMATE_W2EMPL_BRW);
        fseek(reptr, 0, 0);

        /* initialize the number of queues (one queue per gram) and number */
        /* of offset locations for the grams
        numqueues = 0;
        numdups = 0;
        test_info->grprune = 0;

        /* open the file for the ssn duplicate file
        concat(trash, filename, ".ssn.dup");
        ssndup = fopen(trash, "r");
        if(ssndup == NULL)
            return(-1);

        /* open the file for the ssn b+ tree
        concat(trash, filename, ".ssn.idx");
        ssnptr = fopen(trash, "r");
        if(ssnptr == NULL)
            return(-1);

        /* open the file for the name duplicate file
        concat(trash, filename, ".name.dup");
        namedup = fopen(trash, "r");
        if(namedup == NULL)
            return(-1);

        /* open the file for the name b+ tree
        concat(trash, filename, ".name.idx");
        indxptr = fopen(trash, "r");
        if(indxptr == NULL)
            return(-1);

        /* open the file for the name duplicate file
        concat(trash, filename, ".name.dup");
        namedup = fopen(trash, "r");
        if(namedup == NULL)
            return(-1);

        /* open the file for the name b+ tree
        concat(trash, filename, ".name.idx");
        indxptr = fopen(trash, "r");
        if(indxptr == NULL)
            return(-1);

        /* open the duplicate file and read in the queue of matches,
        /* return the total number of matches to the gram

```



```

if(locations[i][locindex[i]].dupe_offset > maxoffset)
{
    maxoffset = locations[i][locindex[i]].dupe_offset;
}
}

/* If the maximum offset is not -1 (indicating all done), then ... */
if(maxoffset != -1)
{
    /* for each of the queues, check to see if it contains the max
    /* offset. If it does, then increment the count of offsets, and
    /* "pop" it from the queue
    for(i = 0; i < numqueues; i++)
    {
        if(locations[i][locindex[i]].dupe_offset == maxoffset)
        {
            if(weight == 0)
                recordweight = locations[i][locindex[i]].dupe_weight;
            weight++;
            /* remove from queue (i any duplicates) - these will only muddy
            /* the count. When a record has more than one of the same gram
            /* it will either be picked up in another queue (the gram was
            /* in input record twice and therefore two queues were set up
            /* for it), or the gram was only in the input record once, and
            /* if it is in the offset record more than once it should not
            /* add it to it's priority because it will give it a false
            /* higher priority
            while(locations[i][locindex[i]].dupe_offset == maxoffset)
                locindex[i]++;
        }
    }

    if((ssn_entered) && (name_entered))
        recordweight = recordweight + 3;

    recount++;
    if((DICE == TRUE) && (name_entered) && (weight > 1) )
    {
        dice=((double)(2*weight)/(double)(numqueues+recordweight))*DICE_SCALE;
        weight = (int) dice;
    }
    else
        weight = 0;

    /* If priority queue array is full (at max memory allocated) then
    /* realloc more memory a little at a time. This will help to
    /* regulate the amount of memory allocated
    if((PRUNE == TRUE) && (weight < PRUNE_WEIGHT))
        pr_count++;
    else
    {
        /* bound the weight at either ends of the array
        if(weight == 0)
            weight = 1;
        if(weight > DICE_GRAIN)
            weight = DICE_GRAIN;

        if(sortindex[weight-1] == maxindex[weight-1])
        {
            maxindex[weight-1] = maxindex[weight-1] + REALLOCFACOR;
            sortloc[weight-1] = (long *) realloc(sortloc[weight-1],
                maxindex[weight-1] * sizeof(long));
        }
    }
}

/* place the offset into the queue matching its priority (# grams */
/* is equal to the array index) and increment the index value of */
/* the queue
sortloc[weight-1][sortindex[weight-1]] = maxoffset;
sortindex[weight-1]++;
}

while(maxoffset != -1); /* end of do ... while loop
test_info->recspruned = pr_count;
return(reccount);
}

/* get dupe queue()
* this function will read from the duplicate file a list of offsets
* which match a given gram
* Input: location of the list in the duplicate file, dupe_loc
* pointer to beginning of the duplicate queue, queue_start
* file pointer to the duplicate file
* Output: memory is allocated for the duplicate queues, the file
* is read for the queues and they are stored in the
* allocated arrays, and a -1 is placed at the end of the list
*/

get_dupe_queue(dupe_loc, queue_start, count, dupe_file, test_info)
long dupe_loc, count;
struct RECINFO **queue_start;
FILE *dupe_file;
struct TEST_DATA *test_info;
{
    /* seek to the location in the duplicate file of the list of offsets */
    fseek(dupe_file, dupe_loc, 0);

    /* allocate memory to hold "count" offsets
    *queue_start=(struct RECINFO *)calloc(count+1, sizeof(struct RECINFO));
    if(*queue_start == NULL)
        return(-1);

    test_info->mem = test_info->mem + ((count+1) * sizeof(struct RECINFO));
    /* read in the queue of data (all at once to minimize disk access) */
    if(fread(*queue_start, sizeof(struct RECINFO), count, dupe_file) == 0)
        return(-1);

    /* append a -1 to the end of the queue
    (*queue_start)[count].dupe_offset = -1;
    return(TRUE);
}

```



```

#
# Bottom Level Makefile (~/.ssapilot/src/bln/sysadm_print)
#
# This make file is at the lowest level in the
# project heirarchy. It is used to actually
# compile, install, clean or wipe bare the
# source directory and associated files in
# the binary directory. It will also compile
# a list of file dependencies for the source files.
#
# This is a list of the key directories in the
# project heirarchy -- the root directory, the
# library directory, the include directory, and
# the binary directory
PROJECT_ROOT = ../..
LIBDIR      = $(PROJECT_ROOT)/lib
INCDIR      = $(PROJECT_ROOT)/include
BINDIR      = $(PROJECT_ROOT)/bin

# This is a list of the key filenames in the
# project -- the executable, the source files,
# the header files, the libraries, the linker
# line for the libraries, the object files,
# the compile flags and the compiler command
EXECUTABLE  = sysadm_print
SRC         = sysadm_print.c
LIBS        = $(LIBDIR)/libgen.oamete.a
CLIBS       = -lgen.oamete -lm
OBJ         = sysadm_print.o
CFLAGS      = -I$(INCDIR) -L$(LIBDIR)
CC          = cc

# this make directive actually compiles the
# source files to executables
it : $(EXECUTABLE)

# this make directive will compile the source
# files to executables, and copy the files
# to the binary directory
install : $(BINDIR)/$(EXECUTABLE)

# this make directive will remove all the
# object files from the source directory
clean :
    rm -f $(OBJ)

# this make directive will remove all of
# the files which can be remade from the
# source directory and the binary directories
bare : clean
    rm -f $(EXECUTABLE)
    rm -f $(BINDIR)/$(EXECUTABLE)

# this make directive will compile a list of
# dependencies for each of the source files
depend : $(SRC)
    sed -e '1,/^\#\# DO NOT DELETE/Id' Makefile > mm.tmp
    cat dependlist >> mm.tmp
    mv mm.tmp Makefile.bak
    mv mm.tmp Makefile
    rm -f dependlist

# directive for the executable
$(EXECUTABLE) : $(OBJ) $(LIBS)
    $(CC) $(OBJ) $(CLIBS) $(CFLAGS) -o $(EXECUTABLE)

# directive for the executable in the binary directory
$(BINDIR)/$(EXECUTABLE) : $(EXECUTABLE)
    cp $(EXECUTABLE) $(BINDIR)

.c.o :
    $(CC) -c $(CFLAGS) $<

# DO NOT DELETE THIS LINE - make depend uses it
search_print.o: search_print.c
search_print.o: /usr/include/stdio.h
search_print.o: /usr/include/stdlib.h
search_print.o: /usr/include/sys/types.h
search_print.o: /usr/include/sys/select.h
search_print.o: /usr/include/sys/time.h
search_print.o: /usr/include/sys/types.h
search_print.o: /usr/include/time.h
search_print.o: /usr/include/sys/sysmacros.h
search_print.o: /usr/include/sys/time.h
search_print.o: /usr/include/malloc.h
search_print.o: /usr/include/math.h
search_print.o: /usr/include/floatnpoint.h
search_print.o: /usr/include/sys/leefp.h
search_print.o: /usr/include/fcntl.h
search_print.o: /usr/include/rpc/rpc.h
search_print.o: /usr/include/rpc/types.h
search_print.o: /usr/include/sys/types.h
search_print.o: /usr/include/sys/time.h
search_print.o: /usr/include/tuser.h
search_print.o: /usr/include/sys/tuser.h
search_print.o: /usr/include/fcntl.h
search_print.o: /usr/include/memory.h
search_print.o: /usr/include/rpc/xdr.h
search_print.o: /usr/include/sys/byteorder.h
search_print.o: /usr/include/sys/types.h
search_print.o: /usr/include/stdio.h
search_print.o: /usr/include/rpc/auth.h
search_print.o: /usr/include/rpc/xdr.h
search_print.o: /usr/include/sys/cred.h
search_print.o: /usr/include/sys/t_lock.h
search_print.o: /usr/include/sys/machlock.h
search_print.o: /usr/include/sys/types.h
search_print.o: /usr/include/sys/dk1_kinfo.h
search_print.o: /usr/include/sys/types.h
search_print.o: /usr/include/sys/sleepq.h
search_print.o: /usr/include/sys/turnstile.h
search_print.o: /usr/include/sys/types.h
search_print.o: /usr/include/sys/param.h
search_print.o: /usr/include/limits.h
search_print.o: /usr/include/unistd.h
search_print.o: /usr/include/sys/fcntl.h
search_print.o: /usr/include/sys/signal.h
search_print.o: /usr/include/vm/faultcode.h
search_print.o: /usr/include/sys/types.h
search_print.o: /usr/include/sys/proc.h
search_print.o: /usr/include/sys/sleepq.h
search_print.o: /usr/include/sys/mutex.h
search_print.o: /usr/include/sys/types.h
search_print.o: /usr/include/sys/machlock.h
search_print.o: /usr/include/sys/turnstile.h

```

```
search_print.o: /usr/include/sys/dk_lkinfo.h
search_print.o: /usr/include/rpc/clnt.h
search_print.o: /usr/include/rpc/rpc_com.h
search_print.o: /usr/include/sys/netconfig.h
search_print.o: /usr/include/rpc/rpc_msg.h
search_print.o: /usr/include/rpc/clnt.h
search_print.o: /usr/include/rpc/auth_sys.h
search_print.o: /usr/include/rpc/auth_des.h
search_print.o: /usr/include/rpc/auth_kerb.h
search_print.o: /usr/include/kerberos/krb.h
search_print.o: /usr/include/kerberos/mit-copyright.h
search_print.o: /usr/include/kerberos/des.h
search_print.o: /usr/include/kerberos/mit-copyright.h
search_print.o: /usr/include/sys/socket.h
search_print.o: /usr/include/sys/netconfig.h
search_print.o: /usr/include/netinet/in.h
search_print.o: /usr/include/sys/stream.h
search_print.o: /usr/include/sys/vnode.h
search_print.o: /usr/ucbinclude/sys/types.h
search_print.o: /usr/include/sys/t_lock.h
search_print.o: /usr/include/sys/time.h
search_print.o: /usr/include/sys/cred.h
search_print.o: /usr/include/sys/uio.h
search_print.o: /usr/ucbinclude/sys/types.h
search_print.o: /usr/include/sys/poll.h
search_print.o: /usr/include/sys/strmdep.h
search_print.o: /usr/include/sys/cred.h
search_print.o: /usr/include/sys/t_lock.h
search_print.o: /usr/include/sys/byteorder.h
search_print.o: /usr/include/rpc/svc.h
search_print.o: /usr/include/rpc/rpc_com.h
search_print.o: /usr/include/rpc/rpc_msg.h
search_print.o: /usr/include/rpc/svc_auth.h
search_print.o: /usr/include/rpc/svc.h
search_print.o: /usr/include/rpc/rpcb_clnt.h
search_print.o: /usr/include/rpc/rpcb.h
search_print.o: /usr/include/rpc/rpcb_prot.h
search_print.o: /usr/include/rpc/rpc.h
search_print.o: ../../include/params.h
search_print.o: ../../include/eamatestruct.h
```





```

case MATE_W2FT: /* Final total Record
    fread(&final, sizeof(struct EAMATE_W2FINAL_TOT), 1, data);
    print_final_rec(final, testfile);
    Index3++;
    break;
}

fclose(testfile);
return(1);
}

/*
*****
* This record prints an employer header record on the
* screen
*
* Input: structure containing the employer header record
*
* Output: The employer record is printed
*****
*/

print_employee_rec()
{
    print_employee_rec(emp_rec, testfile)
    struct EAMATE_W2EMPLR_HEADER empl_rec;
    FILE *testfile;

    {
        fprintf(testfile, "\n"); /* Go to a new page */
        fprintf(testfile, "EIN %s", empl_rec.ein); /* Emplr Num */
        fprintf(testfile, "EST %s", empl_rec.est); /* EST */
        fprintf(testfile, "RPT-YR %s", empl_rec.rpt_yr); /* Year of Report */
        fprintf(testfile, "PRC-YR %s", empl_rec.prc_yr); /* Yr Processed */
        fprintf(testfile, "TAPE-LIB-NO %s", empl_rec.tape_lib_num); /* IL # */
        fprintf(testfile, "%s", empl_rec.type_emplr); /* Type of Emplr */
        fprintf(testfile, "%s", empl_rec.name_code); /* Name Code */
        fprintf(testfile, "OTH-EIN %s", empl_rec.other_ein); /* Other EIN */
        fprintf(testfile, "%s", empl_rec.mrn); /* Record Number */
        fprintf(testfile, "%s", empl_rec.name); /* Employee Name */
        fprintf(testfile, "%s", empl_rec.street_add); /* Street Address */
        fprintf(testfile, "%s", empl_rec.city); /* City */
        fprintf(testfile, "%s", empl_rec.state); /* State */
        fprintf(testfile, "%s", empl_rec.zip_code); /* Zip Code */

        fprintf(testfile, "\n");
        fprintf(testfile, "P ANNUAL ANNUAL ANNUAL");
        fprintf(testfile, "ADVANCE MEDICARE\n");
        fprintf(testfile, "SOCIAL FICA FICA WAGES/TIPS");
        fprintf(testfile, "E D FICA FICA EARNED WAGES MEDICARE");
        fprintf(testfile, "SECURITY NO NAME AND ADDRESS");
        fprintf(testfile, "N C WAGES TIPS OTHER-COMP");
        fprintf(testfile, "TAX W/H TAX W/H INCOME AND TIPS TAX");
        fprintf(testfile, "NUMBER\n");
    }

    print_intermed_rec()
    *****
    * This record prints an intermediate total record on the
    * screen
    *
    * Input: structure containing the intermediate total record
    *
    * Output: The intermediate total record is printed
    *****
}

```







## **E Scout Comments from EAMATE Testing Session**

This appendix contains scouts' comments obtained during the third testing session conducted by NIST during January 1993. Over a six and a half day period consisting of five sessions a day, approximately 20 scouts were trained to use the EAMATE prototype system. Training times ranged from a half hour to an hour and a half, and the average training time was an hour. It must be noted that most of the scouts had never used a mouse, and none were familiar with Microsoft Windows. The scouts were absolutely amazed at the response time of the searches and the robustness of the search engine. After learning how to maneuver around the system, the scouts spent their time trying to "fool" the system by entering incorrect names and/or social security numbers. The scouts performed at least 20 searches per hour and were able to locate the required data. The scouts' comments are included below. (NOTE: Comments are paraphrased unless contained in quotes.)

---

"This system will revolutionize scouting. It will eliminate the human error factor caused by fatigue, poor spelling and simple carelessness. The system finds matches to names and SSNs that the scout would never think of, and, it works at a speed that could never be achieved by a human being.

"The whole time I was using the system, I kept thinking about how easy it was. If the person was on the report, you couldn't help but find them! If the person was not on the report, you knew immediately; it greatly reduced the number of records you needed to look at to make that determination.

"If the optical disk EAMATE could be linked to the SER, BAU Report and DECS, we could eliminate so many paper listings. The listings would ALWAYS be up-to-date and the time involved copying, distributing and accessing the listings would be saved. It would also eliminate accessing ERQY in open period cases -- the system takes you to the report without knowing an MRN or tells you if there is no report.

I love optical disk EAMATE!!!!"

---

"Upon examining and accessing the on-line EAMATE prototype, I was quite impressed.

"Every attempt has been made to cover all potential problems/questions that may arise, as well as enhancements which definitely make this program user-friendly.

"It is my belief, that this prototype is the most comprehensive since our sources started being placed on-line."

---

"I think the program set-up was excellent and it was a challenge to work on the PC for the EAMATE pilot." I think it is a good idea for the EAMATE data to be placed on a PC. The system was fast and easy for me to learn. "I love the mouse. The instructors were very good."

---

"I think the sessions were very helpful. I think that the new EAMATE optical disc [project] will be a benefit in the work I do. It covers everything ... The teachers were very helpful. I enjoyed myself."

---

"Working with the EAMATE system will help save time by not having to wait in sometimes long lines in order to get the film. The system really has its advantages. It identifies the employer along with the beginning of the report and the end of the report. It also gives you the amount of employees in the report and it also can search for an employee in a particular report even if the name is spelled wrong or if the SSN is incorrect. Having a system like this is really a good asset and I would love to have this system. One important thing I found out was that use of this system can also cut out the use of the COMET REL, the REL, and the on-line REL."

---

"The optical disc project is one of the best things that could happen to the Claims Coverage Branch." To be able to process cases using the system instead of film is great even if only one year's worth of data is on the system at present. I like the different ways of finding a person, by name and/or social security number. "I like the mouse." I especially like the possibility of being able to work and complete a blanket case from start to finish at one sitting and at one terminal. This feature would help with the current backlog and would be cost effective. "The technical ladies have done a fantastic job. Thanks to them both."

---

"I think the system for the EAMATE prototype is great. This system will save time in searching for employer and employee information. I hope I am able to use this system before I retire. I also think the coordinators did a great job putting the information we gave them [into the system] and coming up with this prototype."

---

This new EAMATE prototype will be a good asset in aiding work performance. It was very easy to find the information that I needed and it will save a lot of time. "I found that even the most difficult names were easy to locate."

---

"The EAMATE optical disk [prototype] is very advantageous because it:

- a) eliminates the possibility that a particular EAMATE film is unavailable in the film files.
- b) saves time [because] it enables us to get the complete report by entering the report year and the EIN (on the microfilm, it is possible to pull a film only to find out that the remainder of the report is on the next box of microfilm).
- c) should lessen the amount of 7010s being written up by the ECT's. Because of the way it



[the prototype] is programmed, there is a far less likelihood of the 'EO' conditions previously made based on the microfilm search.

- d) is still possible to find your W/E [wage earner] even if the incorrect SSN or name is entered.
  - e) allows the scout to make an immediate printout of individual W/E's or the ER processing information (total EEs, wages, tax w/h, etc)."
- 

"I've always been willing to change for the best. EAMATE on the computer will certainly trim scouting time from my workload. I found the EAMATE pilot training session to be interesting and informative. I couldn't find anything missing from the various screens that I would need during the scouting procedure. I felt that learning to use the mouse attachment came easily enough to me. Personally, I can't wait for the implementation of EAMATE on the computer."

---

"This machine is just too smart! [EAMATE online] will be very effective in processing work. You are able to locate an employee even if the name has been misspelled or transposed, or [if an] incorrect SSN [is] entered. Employees who enter information incorrectly or by mistake will not have to be concerned that they have done something wrong. This system will find the correct employee for you."

---

The EAMATE optical disk will be an asset to the scouts by allowing faster and more efficient processing as compared to the current scouting procedure. "The method of finding employees is remarkable and unique. The representatives from NIST were excellent in explaining and demonstrating the machine."

---

"The EAMATE optical disk pilot was very interesting. The speed and clarity of the system provided much more information [compared to the current system]. For example, using the SSN order format provided extra data that could have possibly been overlooked doing routine scouting. The risk of errors is small. Built-in safeguards eliminate that problem. The system is user-friendly and very efficient."

---

"The EAMATE optical disk pilot has proven, through testing, to be very efficient."

---

Use of the EAMATE on-line should be very time-saving and precise. The pilot showed how effective this system will be in locating wage earners within [a] limited time, which in retrospect will allow more work to be done in less time. The EAMATE system will also enable you to find needed information even though very little information has been provided regarding the information request.

It is my opinion that implementation of EAMATE on-line, is well worth it."

---



"I really enjoyed having the opportunity to participate in the EAMATE optical disk pilot.

"It would be a very useful tool as far as people searching or scouting employee wages, social security numbers, or names. We did a lot of changing names and social security numbers and most of the time the information would be in the first ten names. If incorrect information was sent in by the employer, the machine would find it and give you the correct information. The instructors were very informative and comfortable with the EAMATE optical disk. I hope this [system] will be implemented before I retire."

---

"The microfilm information on the optical disk is put together in an outstanding way. I enjoyed using the mouse, instead of the enter key.

"My being a part of the training was a new learning experience that would be valuable in processing the work in my area [which is the blanket unit]."

## F. Preliminary Employer Report Statistics

The following sections contain some preliminary employer report statistics which were generated during the laboratory experiments performed at NIST on the prototype system. These statistics will be much expanded when SSA sets up the expanded prototype at OCRO in Baltimore, Maryland. However, these preliminary statistics may be useful in determining the amount of disk space, memory, and time necessary to convert the employer reports for SSA's system. In particular, the Txt/COM and the Brw/COM ratio columns will be very helpful in the parse process in order to determine browse and detail locations.

### F.1 Einstats File

EIN	SEQ	Num Recs	Word Count	Avg WCSIZE	Txt Size	COM Size	Txt/ COM	Brw Size	Brw/ COM	BLoc	DLoc
39-1390489	AAA	29	59	2.034	8.64	15.33	0.564	2.83	0.185	1	1
22-2761297	AAA	5266	10935	2.077	1462.03	2522.45	0.580	514.26	0.204	1	1
56-1336585	AAA	501	1030	2.056	139.75	242.23	0.577	48.93	0.202	1	1
51-0331454	AAA	3973	8936	2.249	1103.09	1901.87	0.580	387.99	0.204	1	1
71-0517471	AAA	3971	9246	2.328	1102.55	1901.09	0.580	387.79	0.204	1	1
35-1152814	AAA	505	1131	2.240	140.83	243.79	0.578	49.32	0.202	1	1
35-1152814	AAB	500	1092	2.184	139.48	241.19	0.578	48.83	0.202	1	1
95-6000827	AAA	500	1033	2.066	139.48	241.19	0.578	48.83	0.202	1	1
22-6001794	AAA	500	1056	2.112	139.48	241.19	0.578	48.83	0.202	1	1
94-0902780	AAA	501	1025	2.046	139.75	242.23	0.577	48.93	0.202	1	1
95-1699855	AAA	501	1023	2.042	139.75	242.23	0.577	48.93	0.202	1	1
04-1666290	AAA	501	1037	2.070	139.75	242.23	0.577	48.93	0.202	1	1
01-0212444	AAA	501	1029	2.054	139.75	242.23	0.577	48.93	0.202	1	1
59-2594590	AAA	500	1023	2.046	139.48	241.19	0.578	48.83	0.202	1	1
59-2594590	AAB	94	188	2.000	26.78	47.28	0.566	9.18	0.194	1	1
91-1484586	AAA	250	533	2.132	70.12	121.96	0.575	24.41	0.200	1	1
62-1091294	AAA	500	1013	2.026	139.48	241.19	0.578	48.83	0.202	1	1
34-4431174	AAA	500	1018	2.036	139.48	241.19	0.578	48.83	0.202	1	1
36-2708624	AAA	250	513	2.052	70.12	121.96	0.575	24.41	0.200	1	1
54-6001718	AAA	4009	7377	1.840	1113.12	1919.54	0.580	391.50	0.204	1	1
34-0846262	AAA	3984	8143	2.044	1106.33	1907.85	0.580	389.06	0.204	1	1
13-3591193	AAA	3999	9860	2.466	1110.40	1914.99	0.580	390.53	0.204	1	1
71-0225020	AAA	4010	8982	2.240	1113.39	1919.93	0.580	391.60	0.204	1	1
54-0332635	AAA	4011	8137	2.029	1113.66	1920.32	0.580	391.70	0.204	1	1
13-2721761	AAA	4014	8187	2.040	1114.47	1921.49	0.580	391.99	0.204	1	1
54-6001128	AAA	4026	10090	2.506	1117.98	1927.85	0.580	393.16	0.204	2	1
93-0654045	AAA	250	504	2.016	70.12	121.96	0.575	24.41	0.200	2	1
74-1109737	AAA	4027	8102	2.012	1118.25	1928.89	0.580	393.26	0.204	2	1
25-1581273	AAA	1002	2090	2.086	278.73	481.09	0.579	97.85	0.203	2	1
39-0379990	AAA	4025	8152	2.025	1117.71	1927.46	0.580	393.07	0.204	2	1
56-0348490	AAA	3973	8692	2.188	1103.09	1901.87	0.580	387.99	0.204	2	1
59-6000396	AAA	4023	8357	2.077	1117.16	1926.68	0.580	392.87	0.204	2	1
14-1588290	AAA	250	518	2.072	70.12	121.96	0.575	24.41	0.200	2	1
47-0524851	AAA	1003	2125	2.119	279.00	481.48	0.579	97.95	0.203	2	1
31-0859857	AAA	1002	2203	2.199	278.73	481.09	0.579	97.85	0.203	2	1
25-6000141	AAA	1002	2077	2.073	278.73	481.09	0.579	97.85	0.203	2	1



86-0410573	AAA	250	505	2.020	70.12	121.96	0.575	24.41	0.200	2	1
21-0635010	AAA	1002	2137	2.133	278.73	481.09	0.579	97.85	0.203	2	1
75-2257592	AAA	1002	2034	2.030	278.73	481.09	0.579	97.85	0.203	2	1
41-0719940	AAA	1002	2044	2.040	278.73	481.09	0.579	97.85	0.203	2	1
14-1682942	AAA	2499	5180	2.073	694.03	1196.61	0.580	244.04	0.204	2	1
85-6005550	AAA	2500	5167	2.067	694.30	1197.00	0.580	244.14	0.204	2	1
62-0400610	AAA	2501	5075	2.029	694.58	1197.39	0.580	244.24	0.204	2	1
94-0472650	AAA	2505	5122	2.045	695.91	1200.64	0.580	244.63	0.204	2	1
73-0579283	AAA	2494	5089	2.040	692.67	1194.66	0.580	243.55	0.204	2	1
91-0984002	AAA	501	1029	2.054	139.75	242.23	0.577	48.93	0.202	2	1
38-6006063	AAA	2499	5127	2.052	694.03	1196.61	0.580	244.04	0.204	2	1
06-1204645	AAA	250	505	2.020	70.12	121.96	0.575	24.41	0.200	2	1
54-6001803	AAA	1000	1093	1.093	278.18	480.31	0.579	97.66	0.203	2	1
15-0543659	AAA	2499	5164	2.066	694.03	1196.61	0.580	244.04	0.204	2	1
43-0977042	AAA	1001	2046	2.044	278.45	480.70	0.579	97.75	0.203	2	1
56-0792028	AAA	999	2132	2.134	277.91	479.92	0.579	97.56	0.203	2	1
59-0836811	AAA	1001	2162	2.160	278.45	480.70	0.579	97.75	0.203	2	1
71-0480157	AAA	1001	2135	2.133	278.45	480.70	0.579	97.75	0.203	2	1
36-2930831	AAA	501	1028	2.052	139.75	242.23	0.577	48.93	0.202	2	1
75-0755367	AAA	1002	2015	2.011	278.73	481.09	0.579	97.85	0.203	2	1
11-1001790	AAA	1000	2047	2.047	278.18	480.31	0.579	97.66	0.203	2	1
23-1352620	AAA	1000	2100	2.100	278.18	480.31	0.579	97.66	0.203	2	1
54-1549763	AAA	1002	1834	1.830	278.73	481.09	0.579	97.85	0.203	1	2
74-1659656	AAA	4009	4509	1.125	1113.12	1919.54	0.580	391.50	0.204	1	2
38-2113393	AAA	2018	4080	2.022	560.73	967.89	0.579	197.07	0.204	1	2
38-2113393	AAB	50	104	2.080	14.59	26.50	0.551	4.88	0.184	1	2
38-2113393	AAC	1453	2938	2.022	403.88	696.82	0.580	141.89	0.204	1	2
38-2113393	AAD	163	330	2.025	45.76	79.10	0.579	15.92	0.201	1	2
38-2113393	AAE	1110	2259	2.035	308.79	533.69	0.579	108.40	0.203	1	2
38-2113393	AAF	482	988	2.050	134.34	232.49	0.578	47.07	0.202	1	2
38-2113393	AAG	92	185	2.011	26.73	51.17	0.522	8.98	0.176	1	2
38-2113393	AAH	2509	5117	2.039	697.49	1210.12	0.576	245.02	0.202	1	2
38-2113393	AAI	105	109	1.038	29.77	52.21	0.570	10.25	0.196	1	2
38-2113393	AAJ	693	1403	2.025	192.86	332.89	0.579	67.68	0.203	1	2
38-2113393	AAK	758	1546	2.040	211.00	364.84	0.578	74.02	0.203	1	2
38-2113393	AAL	2827	2955	1.045	785.06	1353.51	0.580	276.07	0.204	1	2
38-2113393	AAM	390	397	1.018	108.87	188.72	0.577	38.09	0.202	1	2
38-2113393	AAN	15	15	1.000	4.84	9.22	0.525	1.46	0.159	1	2
38-2113393	AAO	344	355	1.032	96.14	166.51	0.577	33.59	0.202	1	2
38-2113393	AAP	4941	5058	1.024	1371.82	2365.69	0.580	482.52	0.204	1	2
38-2113393	AAQ	1647	3322	2.017	457.79	790.21	0.579	160.84	0.204	1	2
38-2113393	AAR	26	52	2.000	7.83	14.16	0.553	2.54	0.179	1	2
38-2113393	AAS	3387	3519	1.039	940.55	1622.24	0.580	330.76	0.204	1	2
38-2113393	AAT	999	2008	2.010	277.91	479.92	0.579	97.56	0.203	1	2
38-2113393	AAU	677	1368	2.021	188.52	326.01	0.578	66.11	0.203	1	2
38-2113393	AAV	2049	4135	2.018	569.15	981.26	0.580	200.10	0.204	1	2
38-2113393	AAW	44	92	2.091	12.96	23.51	0.551	4.30	0.183	1	2
58-1516994	AAA	2494	5381	2.158	692.67	1194.66	0.580	243.55	0.204	1	2
44-6000524	AAA	2488	5073	2.039	691.05	1191.67	0.580	242.97	0.204	1	2
33-0363116	AAA	3991	8102	2.030	1108.23	1911.23	0.580	389.75	0.204	1	2
95-6042622	AAA	1003	2118	2.112	279.00	481.48	0.579	97.95	0.203	1	2
76-0146568	AAA	1003	2041	2.035	279.00	481.48	0.579	97.95	0.203	1	2
75-2255014	AAA	3996	8232	2.060	1109.59	1913.17	0.580	390.23	0.204	1	2
33-0363116	AAB	3991	8102	2.030	1108.23	1911.23	0.580	389.75	0.204	1	2



13-6400571	AAA	19804	40226	2.031	5496.58	9475.86	0.580	1933.98	0.204	1	2
68-0038644	AAA	1002	2101	2.097	278.73	481.09	0.579	97.85	0.203	1	2
94-3041767	AAA	19921	22138	1.111	5528.84	9530.67	0.580	1945.41	0.204	1	2
87-6000525	AAA	20035	43453	2.169	5560.53	9585.35	0.580	1956.54	0.204	1	2
04-2103594	AAA	19883	41860	2.105	5518.27	9512.23	0.580	1941.70	0.204	1	2
13-5218870	AAA	20132	20674	1.027	5587.60	9632.76	0.580	1966.02	0.204	1	2
52-6000989	AAA	20166	41102	2.038	5596.83	9647.96	0.580	1969.34	0.204	1	2
41-0749934	AAA	20174	40613	2.013	5599.50	9655.75	0.580	1970.12	0.204	1	2
76-0178498	AAA	20235	44500	2.199	5616.06	9681.46	0.580	1976.07	0.204	1	2
13-6295168	AAA	20314	21175	1.042	5638.00	9718.87	0.580	1983.79	0.204	1	2
74-2248983	AAA	30024	30563	1.018	8332.67	14364.2	0.580	2932.03	0.204	1	2
42-6004813	AAA	30186	62623	2.075	8377.64	14441.9	0.580	2947.85	0.204	1	2
63-1032906	AAA	2499	5346	2.139	694.03	1196.61	0.580	244.04	0.204	1	2
23-1989084	AAA	30331	32358	1.067	8417.75	14510.6	0.580	2962.01	0.204	1	2
13-3354541	AAA	30371	31195	1.027	8428.85	14529.8	0.580	2965.92	0.204	1	2
63-0941966	AAA	39669	84415	2.128	11009.2	18977.7	0.580	3873.93	0.204	1	2
31-1153510	AAA	39895	40333	1.011	11072.0	19086.2	0.580	3896.00	0.204	1	2
41-0901437	AAA	41426	83894	2.025	11496.8	19818.0	0.580	4045.51	0.204	1	2
36-1115800	AAA	40471	41879	1.035	11231.8	19361.6	0.580	3952.25	0.204	1	2
93-0432081	AAA	48327	99317	2.055	13411.8	23118.6	0.580	4719.43	0.204	1	2
36-1282500	AAA	49059	99956	2.037	13615.2	23469.3	0.580	4790.92	0.204	1	2
38-6006309	AAA	51709	110246	2.132	14422.1	25120.7	0.574	5049.71	0.201	1	2
04-3008884	AAA	71570	144306	2.016	19862.0	34237.2	0.580	6989.26	0.204	1	2
86-0096778	AAA	500	1038	2.076	139.48	241.19	0.578	48.83	0.202	1	2
13-5318100	AAA	76672	163660	2.135	21278.0	36678.5	0.580	7487.50	0.204	2	2
38-1274536	AAA	77981	156611	2.008	21641.0	37303.2	0.580	7615.33	0.204	2	2
71-0427007	AAA	79861	83475	1.045	22162.8	38203.0	0.580	7798.93	0.204	2	2
59-0248365	AAA	4943	9896	2.002	1373.11	2377.38	0.578	482.71	0.203	2	2
61-0144470	AAA	4947	10048	2.031	1373.45	2368.02	0.580	483.11	0.204	2	2
61-0144470	AAB	121	271	2.240	34.11	59.10	0.577	11.82	0.200	2	2
63-6000997	AAA	4952	10128	2.045	1374.81	2370.62	0.580	483.59	0.204	2	2
77-0098061	AAA	4953	10167	2.053	1375.08	2371.01	0.580	483.69	0.204	2	2
13-3106295	AAA	4974	5142	1.034	1381.03	2381.53	0.580	485.74	0.204	2	2
61-1028725	AAA	4981	10070	2.022	1382.93	2384.91	0.580	486.43	0.204	2	2
36-1717960	AAA	5065	10374	2.048	1406.23	2424.91	0.580	494.63	0.204	2	2
87-0453008	AAA	2494	5097	2.044	692.67	1194.66	0.580	243.55	0.204	2	2
74-0484030	AAA	10137	13573	1.339	2813.83	4851.51	0.580	989.94	0.204	2	2
04-1734655	AAA	10117	20848	2.061	2808.16	4841.38	0.580	987.99	0.204	2	2
63-6001138	AAA	10114	20703	2.047	2807.34	4839.56	0.580	987.70	0.204	2	2
41-1260605	AAA	10046	21404	2.131	2788.63	4808.13	0.580	981.05	0.204	2	2
86-0414274	AAA	10017	20269	2.023	2780.51	4793.84	0.580	978.22	0.204	2	2
63-0857290	AAA	10035	19743	1.967	2785.40	4802.16	0.580	979.98	0.204	2	2
98-0018947	AAA	9979	21411	2.146	2769.95	4775.40	0.580	974.51	0.204	2	2
38-1709248	AAA	9949	20116	2.022	2761.56	4760.72	0.580	971.58	0.204	2	2
74-1586031	AAA	5067	10623	2.097	1406.77	2425.69	0.580	494.82	0.204	2	2
62-0450611	AAA	5067	10932	2.157	1406.77	2425.69	0.580	494.82	0.204	2	2
75-1232789	AAA	123565	251496	2.035	34291.1	59122.5	0.580	12066.8	0.204	1	1
94-1461843	AAA	2509	5155	2.055	696.99	1202.20	0.580	245.02	0.204	1	1
39-0967678	AAA	5018	10092	2.011	1393.22	2402.31	0.580	490.04	0.204	1	1
23-0671120	AAA	5034	10208	2.028	1397.56	2409.85	0.580	491.60	0.204	1	1
15-0398550	AAA	5060	10558	2.087	1404.87	2422.31	0.580	494.14	0.204	1	1
36-2179782	AAA	5050	10140	2.008	1402.15	2417.77	0.580	493.16	0.204	1	1
05-0447226	AAA	5048	10335	2.047	1401.61	2416.99	0.580	492.97	0.204	1	1
76-0264097	AAA	5049	10337	2.047	1401.88	2417.38	0.580	493.07	0.204	1	1

74-1109741	AAA	5012	10347	2.064	1391.59	2399.97	0.580	489.45	0.204	1	1
23-1943113	AAA	5010	10243	2.045	1391.05	2398.55	0.580	489.26	0.204	1	1
38-2072341	AAA	1001	2020	2.018	278.45	480.70	0.579	97.75	0.203	1	1
13-1675535	AAA	500	1020	2.040	139.48	241.19	0.578	48.83	0.202	1	1
31-0559589	AAA	5051	10223	2.024	1402.43	2418.16	0.580	493.26	0.204	1	1
57-0736794	AAA	499	1019	2.042	139.20	240.80	0.578	48.73	0.202	1	1
23-1979193	AAA	501	1047	2.090	139.75	242.23	0.577	48.93	0.202	1	1
53-0204616	AAA	1003	1048	1.045	279.00	481.48	0.579	97.95	0.203	1	1
61-1078924	AAA	499	1052	2.108	139.20	240.80	0.578	48.73	0.202	1	1
31-1045381	AAA	500	1120	2.240	139.48	241.19	0.578	48.83	0.202	1	1
38-6029206	AAA	4990	9972	1.998	1385.37	2388.42	0.580	487.30	0.204	2	1
59-1672120	AAA	2509	5134	2.046	696.99	1202.20	0.580	245.02	0.204	2	1
56-1381211	AAA	2499	5266	2.107	694.03	1196.61	0.580	244.04	0.204	2	1
63-0479282	AAA	3996	8111	2.030	1109.59	1913.17	0.580	390.23	0.204	2	1
59-1698006	AAA	1245	2545	2.044	346.18	597.59	0.579	121.58	0.203	2	1
59-1698006	AAB	8	17	2.125	2.94	5.84	0.503	0.78	0.134	2	1
59-1698006	AAC	371	766	2.065	103.71	180.02	0.576	36.23	0.201	2	1
59-1698006	AAD	1650	3370	2.042	458.60	791.77	0.579	161.13	0.204	2	1
59-1698006	AAE	501	1226	2.447	139.75	242.23	0.577	48.93	0.202	2	1
59-1698006	AAF	4426	9119	2.060	1228.79	2118.39	0.580	432.23	0.204	2	1
22-1801227	AAA	1000	2045	2.045	278.18	480.31	0.579	97.66	0.203	2	1
62-1178938	AAA	998	2296	2.301	277.64	479.53	0.579	97.46	0.203	2	1
25-1211482	AAA	1001	2047	2.045	278.45	480.70	0.579	97.75	0.203	2	1
54-6001471	AAA	500	1204	2.408	139.48	241.19	0.578	48.83	0.202	2	1
56-0940643	AAA	1475	3035	2.058	409.86	706.69	0.580	144.04	0.204	2	1
56-0940643	AAB	2505	5077	2.027	695.91	1200.64	0.580	244.63	0.204	2	1
58-0978843	AAA	500	1063	2.126	139.48	241.19	0.578	48.83	0.202	2	1
23-1691403	AAA	500	1020	2.040	139.48	241.19	0.578	48.83	0.202	2	1
23-1668435	AAA	500	1032	2.064	139.48	241.19	0.578	48.83	0.202	2	1
34-4428218	AAA	997	2027	2.033	277.37	479.14	0.579	97.36	0.203	2	1
35-1797634	AAA	500	1016	2.032	139.48	241.19	0.578	48.83	0.202	2	1
11-2834450	AAA	250	532	2.128	70.12	121.96	0.575	24.41	0.200	2	1
13-3173586	AAA	500	1041	2.082	139.48	241.19	0.578	48.83	0.202	2	1
94-2290265	AAA	250	541	2.164	70.12	121.96	0.575	24.41	0.200	2	1
58-6000865	AAA	250	589	2.356	70.12	121.96	0.575	24.41	0.200	2	1
74-2404626	AAA	999	2094	2.096	277.91	479.92	0.579	97.56	0.203	2	1
23-2578152	AAA	999	1041	1.042	277.91	479.92	0.579	97.56	0.203	2	1
61-0592866	AAA	250	579	2.316	70.12	121.96	0.575	24.41	0.200	2	1
22-2341770	AAA	1001	2176	2.174	278.45	480.70	0.579	97.75	0.203	2	1
52-0886787	AAA	4020	8403	2.090	1116.35	1925.51	0.580	392.58	0.204	2	1
95-3946299	AAA	133	282	2.120	37.62	66.11	0.569	12.99	0.196	2	1
95-3946299	AAB	250	527	2.108	70.12	121.96	0.575	24.41	0.200	2	1
95-0613650	AAA	4989	10095	2.023	1385.10	2388.03	0.580	487.21	0.204	2	1
95-0613650	AAB	14328	29080	2.030	3976.82	6855.47	0.580	1399.22	0.204	2	1
88-0173471	AAA	250	505	2.020	70.12	121.96	0.575	24.41	0.200	2	1
34-1580269	AAA	9968	10555	1.059	2766.96	4770.47	0.580	973.44	0.204	2	1
34-1580269	AAB	5069	5361	1.058	1407.31	2426.47	0.580	495.02	0.204	2	1
22-1211670	AAA	7902	16323	2.066	2193.48	3781.28	0.580	771.68	0.204	1	2
22-1211670	AAB	6144	12482	2.032	1705.58	2940.55	0.580	600.00	0.204	1	2
22-1211670	AAC	3582	7281	2.033	994.72	1715.49	0.580	349.80	0.204	1	2
22-1211670	AAD	101	205	2.030	28.68	50.00	0.574	9.86	0.197	1	2
22-1211670	AAE	9309	18813	2.021	2584.10	4455.50	0.580	909.08	0.204	1	2
22-1211670	AAF	1336	2756	2.063	371.38	640.97	0.579	130.47	0.204	1	2
22-1211670	AAG	107	218	2.037	30.31	52.99	0.572	10.45	0.197	1	2



22-1211670	AAH	4990	10059	2.016	1385.37	2388.42	0.580	487.30	0.204	1	2
52-6002033	AAA	123843	252037	2.035	34368.3	59242.1	0.580	12094.0	0.204	1	2
34-9990000	AAA	250001	716875	2.867	69378.5	119589	0.580	24414.1	0.204	1	2
94-1648752	AAA	319	648	2.031	89.10	154.43	0.577	31.15	0.202	1	2
94-1648752	AAB	236	487	2.064	66.07	114.82	0.575	23.05	0.201	1	2
94-1648752	AAC	454	963	2.121	126.74	219.63	0.577	44.34	0.202	1	2
94-1648752	AAD	258	544	2.109	72.29	125.73	0.575	25.20	0.200	1	2
94-1648752	AAE	2332	5693	2.441	647.71	1116.99	0.580	227.73	0.204	1	2
94-1648752	AAF	1000	2077	2.077	278.18	480.31	0.579	97.66	0.203	1	2
94-1648752	AAG	383	794	2.073	106.97	185.34	0.577	37.40	0.202	1	2
94-1648752	AAH	250	533	2.132	70.12	121.96	0.575	24.41	0.200	1	2
94-1648752	AAI	1371	2868	2.092	381.13	657.60	0.580	133.89	0.204	1	2
61-1158928	AAA	250	503	2.012	70.12	121.96	0.575	24.41	0.200	1	2
59-1698006	AAG	1245	2545	2.044	346.18	597.59	0.579	121.58	0.203	2	2
59-1698006	AAH	8	17	2.125	2.94	5.84	0.503	0.78	0.134	2	2
59-1698006	AAI	371	766	2.065	103.71	180.02	0.576	36.23	0.201	2	2
59-1698006	AAJ	1650	3370	2.042	458.60	791.77	0.579	161.13	0.204	2	2
14-1513238	AAA	5009	10176	2.032	1390.78	2398.16	0.580	489.16	0.204	1	2
13-2526632	AAA	2510	5091	2.028	697.27	1203.23	0.579	245.12	0.204	1	2
58-0964286	AAA	3995	4283	1.072	1109.31	1912.78	0.580	390.14	0.204	1	2
31-1112315	AAA	250	290	1.160	70.12	121.96	0.575	24.41	0.200	1	2
43-1236588	AAA	250	510	2.040	70.12	121.96	0.575	24.41	0.200	1	2
72-0992142	AAA	500	1017	2.034	139.48	241.19	0.578	48.83	0.202	1	2
94-2629822	AAA	271324	550280	2.028	75336.8	130392	0.578	26496.4	0.203	1	2
87-6000525	AAB	20035	43453	2.169	5560.53	9585.35	0.580	1956.54	0.204	1	2
31-6000142	AAA	1002	2044	2.040	278.73	481.09	0.579	97.85	0.203	1	2
11-2831424	AAA	998	2054	2.058	277.64	479.53	0.579	97.46	0.203	1	2
38-1798424	AAA	49664	101131	2.036	13782.9	23758.9	0.580	4850.00	0.204	1	2
38-1510762	AAA	375000	385412	1.028	104067	179383	0.580	36621.1	0.204	1	2
22-1446668	AAA	19	43	2.263	5.93	10.78	0.550	1.86	0.172	1	2
22-1446668	AAB	250	540	2.160	70.12	121.96	0.575	24.41	0.200	1	2
74-2355451	AAA	250	536	2.144	70.12	121.96	0.575	24.41	0.200	1	2
94-2131571	AAA	250	531	2.124	70.12	121.96	0.575	24.41	0.200	1	2
63-0570350	AAA	250	504	2.016	70.12	121.96	0.575	24.41	0.200	1	2
38-0281180	AAA	250	517	2.068	70.12	121.96	0.575	24.41	0.200	1	2
05-0394406	AAA	250	518	2.072	70.12	121.96	0.575	24.41	0.200	1	2
91-1043642	AAA	2500	5009	2.004	694.30	1197.00	0.580	244.14	0.204	1	2
74-1750451	AAA	5017	9408	1.875	1392.95	2401.92	0.580	489.94	0.204	2	2
59-2239528	AAA	5018	11525	2.297	1393.22	2402.31	0.580	490.04	0.204	2	2
65-0079270	AAA	500	1031	2.062	139.48	241.19	0.578	48.83	0.202	2	2
41-0848441	AAA	165398	335151	2.026	45900.4	79120.3	0.580	16152.1	0.204	2	2
71-0333344	AAA	250	583	2.332	70.12	121.96	0.575	24.41	0.200	2	2
41-1317059	AAA	2502	5075	2.028	695.09	1199.47	0.580	244.34	0.204	2	2
16-0743084	AAA	500	1027	2.054	139.48	241.19	0.578	48.83	0.202	2	2
36-3515372	AAA	500	1009	2.018	139.48	241.19	0.578	48.83	0.202	2	2
59-2964349	AAA	250	554	2.216	70.12	121.96	0.575	24.41	0.200	2	2
84-9980000	AAA	205891	431898	2.098	57137.5	98489.1	0.580	20106.5	0.204	1	1
84-9990000	AAA	344999	729059	2.113	95741.4	165031	0.580	33691.3	0.204	1	1
71-0415188	AAA	267590	271196	1.013	74259.6	128003	0.580	26131.8	0.204	1	1
53-9990000	AAA	281634	315619	1.121	78157.1	134721	0.580	27503.3	0.204	1	1
35-9990000	AAA	344999	774468	2.245	95741.4	165031	0.580	33691.3	0.204	2	2
35-9990000	AAB	197983	441760	2.231	54942.9	94706.5	0.580	19334.2	0.204	2	2
35-9990000	AAC	115192	267757	2.324	31967.6	55103.9	0.580	11249.2	0.204	2	2
76-0300290	AAA	501	1073	2.142	139.75	242.23	0.577	48.93	0.202	2	2



59-2612918	AAA	500	1022	2.044	139.48	241.19	0.578	48.83	0.202	2	1
38-1369604	AAA	1004	2036	2.028	279.52	484.20	0.577	98.05	0.202	2	1
63-0967763	AAA	250	555	2.220	70.12	121.96	0.575	24.41	0.200	2	1
58-1837116	AAA	250	550	2.200	70.12	121.96	0.575	24.41	0.200	2	1
56-1703216	AAA	250	506	2.024	70.12	121.96	0.575	24.41	0.200	2	1
37-1174785	AAA	250	515	2.060	70.12	121.96	0.575	24.41	0.200	2	1
56-1349827	AAA	129	260	2.016	36.53	63.90	0.572	12.60	0.197	2	1
56-1349827	AAB	176	353	2.006	49.54	86.50	0.573	17.19	0.199	2	1
56-1349827	AAC	250	520	2.080	70.12	121.96	0.575	24.41	0.200	2	1
95-2013660	AAA	250	519	2.076	70.12	121.96	0.575	24.41	0.200	2	1
82-0325203	AAA	250	549	2.196	70.12	121.96	0.575	24.41	0.200	2	1
36-2590063	AAA	250	530	2.120	70.12	121.96	0.575	24.41	0.200	2	1
36-2590063	AAB	455	993	2.182	127.01	220.02	0.577	44.43	0.202	2	1
36-2590063	AAC	47	94	2.000	13.78	24.68	0.558	4.59	0.186	2	1
38-0763360	AAA	250	515	2.060	70.12	121.96	0.575	24.41	0.200	2	1
53-0196597	AAA	500	1068	2.136	139.48	241.19	0.578	48.83	0.202	2	1
74-2545137	AAA	250	527	2.108	70.12	121.96	0.575	24.41	0.200	2	1
59-0751858	AAA	250	527	2.108	70.12	121.96	0.575	24.41	0.200	2	1
94-2507766	AAA	250	542	2.168	70.12	121.96	0.575	24.41	0.200	2	1
04-2156078	AAA	250	530	2.120	70.12	121.96	0.575	24.41	0.200	2	1
72-0741707	AAA	250	522	2.088	70.12	121.96	0.575	24.41	0.200	2	1
33-0195457	AAA	250	514	2.056	70.12	121.96	0.575	24.41	0.200	2	1
95-0831590	AAA	250	513	2.052	70.12	121.96	0.575	24.41	0.200	2	1
84-0602726	AAA	250	566	2.264	70.12	121.96	0.575	24.41	0.200	2	1
39-1143171	AAA	250	511	2.044	70.12	121.96	0.575	24.41	0.200	2	1
61-0997092	AAA	998	1993	1.997	277.64	479.53	0.579	97.46	0.203	2	1
36-3429602	AAA	500	1022	2.044	139.48	241.19	0.578	48.83	0.202	2	1
41-6034144	AAA	500	1076	2.152	139.48	241.19	0.578	48.83	0.202	2	1
03-6000658	AAA	501	1038	2.072	139.75	242.23	0.577	48.93	0.202	2	1
87-0368169	AAA	501	1034	2.064	139.75	242.23	0.577	48.93	0.202	2	1
38-2500514	AAA	996	2048	2.056	277.10	478.75	0.579	97.27	0.203	2	1
65-0012091	AAA	996	2082	2.090	277.10	478.75	0.579	97.27	0.203	2	1
54-6001605	AAA	997	2072	2.078	277.37	479.14	0.579	97.36	0.203	1	1
84-6010331	AAA	998	2023	2.027	277.64	479.53	0.579	97.46	0.203	1	1
76-0276305	AAA	501	1167	2.329	139.75	242.23	0.577	48.93	0.202	1	1
55-0548701	AAA	501	1112	2.220	139.75	242.23	0.577	48.93	0.202	1	1
91-0907451	AAA	250	531	2.124	70.12	121.96	0.575	24.41	0.200	1	1
16-1251902	AAA	501	1017	2.030	139.75	242.23	0.577	48.93	0.202	1	1
57-0874699	AAA	501	1020	2.036	139.75	242.23	0.577	48.93	0.202	1	1
34-0640780	AAA	501	1032	2.060	139.75	242.23	0.577	48.93	0.202	1	1
59-1698006	AAK	501	1049	2.094	139.75	242.23	0.577	48.93	0.202	2	1
59-1698006	AAL	204	419	2.054	57.14	98.71	0.579	19.92	0.202	2	1
59-1698006	AAM	512	1054	2.059	142.73	247.17	0.577	50.00	0.202	2	1
59-1698006	AAN	1022	2117	2.071	284.16	490.18	0.580	99.80	0.204	2	1
59-1698006	AAO	260	535	2.058	72.84	126.51	0.576	25.39	0.201	2	1
76-0100695	AAA	500	1020	2.040	139.48	241.19	0.578	48.83	0.202	1	1
58-6000147	AAA	500	1020	2.040	139.48	241.19	0.578	48.83	0.202	1	1
64-0752022	AAA	500	1055	2.110	139.48	241.19	0.578	48.83	0.202	1	1
94-3070458	AAA	250	526	2.104	70.12	121.96	0.575	24.41	0.200	1	1
38-6004706	AAA	250	518	2.072	70.12	121.96	0.575	24.41	0.200	1	1
04-1689000	AAA	250	513	2.052	70.12	121.96	0.575	24.41	0.200	1	1
84-0997508	AAA	250	501	2.004	70.12	121.96	0.575	24.41	0.200	1	1
95-4172255	AAA	250	536	2.144	70.12	121.96	0.575	24.41	0.200	1	1
52-1661226	AAA	250	542	2.168	70.12	121.96	0.575	24.41	0.200	1	1

94-1675108	AAA	499	1099	2.202	139.20	240.80	0.578	48.73	0.202	1	1
93-0765253	AAA	499	1042	2.088	139.20	240.80	0.578	48.73	0.202	1	1
95-3800369	AAA	499	1022	2.048	139.20	240.80	0.578	48.73	0.202	1	1
95-3537532	AAA	4028	8259	2.050	1118.52	1929.28	0.580	393.36	0.204	1	2
34-0505560	AAA	4028	8461	2.101	1118.52	1929.28	0.580	393.36	0.204	1	2
71-6021158	AAA	4029	8099	2.010	1118.79	1929.67	0.580	393.46	0.204	1	2
84-0377058	AAA	4029	8328	2.067	1118.79	1929.67	0.580	393.46	0.204	1	2
13-1562932	AAA	250	516	2.064	70.12	121.96	0.575	24.41	0.200	1	2
35-6000169	AAA	997	2104	2.110	277.37	479.14	0.579	97.36	0.203	1	2
11-2239919	AAA	998	2075	2.079	277.64	479.53	0.579	97.46	0.203	1	2
22-6001820	AAA	999	2065	2.067	277.91	479.92	0.579	97.56	0.203	1	2
63-0581231	AAA	230	485	2.109	64.45	111.83	0.576	22.46	0.201	1	2
63-0581231	AAB	999	2173	2.175	277.91	479.92	0.579	97.56	0.203	1	2
36-2513909	AAA	2492	5156	2.069	692.13	1193.88	0.580	243.36	0.204	1	2
94-1606174	AAA	2490	5099	2.048	691.59	1192.45	0.580	243.16	0.204	1	2
38-2706314	AAA	998	2026	2.030	277.64	479.53	0.579	97.46	0.203	1	2
41-1243894	AAA	997	2023	2.029	277.37	479.14	0.579	97.36	0.203	1	2
44-0545813	AAA	997	2065	2.071	277.37	479.14	0.579	97.36	0.203	1	2
53-0196580	AAA	3971	8841	2.226	1102.55	1901.09	0.580	387.79	0.204	1	2
41-0760000	AAA	434578	466571	1.074	120600	207928	0.580	42439	0.204	1	1
04-2849911	AAA	250	512	2.048	70.12	121.96	0.575	24.41	0.200	1	1
36-3540022	AAA	248	511	2.060	69.58	121.18	0.574	24.22	0.200	1	1
36-3540022	AAB	1001	2014	2.012	278.45	480.70	0.579	97.75	0.203	1	1
36-3540022	AAC	85	171	2.012	24.34	43.12	0.564	8.30	0.192	1	1
62-0721803	AAA	3983	4334	1.088	1106.06	1907.46	0.580	388.96	0.204	1	1
86-6000547	AAA	2497	5126	2.053	693.49	1195.83	0.580	243.85	0.204	1	1
23-1265004	AAA	1004	2025	2.017	279.27	482.51	0.579	98.05	0.203	1	1
13-6400571	AAB	19804	40226	2.031	5496.58	9475.86	0.580	1933.98	0.204	1	1
95-1978576	AAA	9923	20184	2.034	2754.50	4749.29	0.580	969.04	0.204	1	1
38-6001599	AAA	5058	12165	2.405	1404.33	2421.54	0.580	493.95	0.204	1	1
73-1032203	AAA	2493	5629	2.258	692.40	1194.27	0.580	243.46	0.204	1	1

## F.2 Indexstats File

EIN	Num Recs	CPU Time	Elapsed Time	Temp Space	Name Idx	Name Dup	SSN Idx	SSN Dup
01-0212444	501	1.40	2	0.00	7.87	45.05	23.61	11.74
04-1666290	501	1.33	1	0.00	7.87	43.34	23.61	11.74
04-1689000	250	0.57	0	0.00	7.87	21.39	7.87	5.86
04-2103594	19883	40.15	44	1193.16	7.87	1719.82	188.91	466.01
04-2849911	250	0.60	1	0.00	7.87	21.26	7.87	5.86
04-3008884	71570	113.61	120	6234.08	23.61	6328.98	228.26	1677.42
05-0394406	250	0.57	1	0.00	7.87	22.01	7.87	5.86
05-0447226	5048	13.68	19	132.13	7.87	451.34	70.84	118.31
11-2239919	998	2.95	3	0.00	7.87	87.25	39.36	23.39
11-2831424	998	2.96	4	0.00	7.87	87.52	31.48	23.39
13-1562932	250	0.69	1	0.00	7.87	22.06	7.87	5.86
13-1675535	500	1.54	2	0.00	7.87	44.20	23.61	11.72



13-2526632	2510	6.78	7	16.02	7.87	214.72	55.10	58.83
13-2721761	4014	11.87	13	68.07	7.87	357.27	86.58	94.08
13-3354541	30371	45.92	48	1097.07	7.87	1581.65	204.65	711.82
13-3591193	3999	11.02	12	92.09	7.87	386.69	78.71	93.73
13-5218870	20132	35.11	37	652.64	7.87	1139.92	181.04	471.84
13-6295168	20314	33.58	35	540.53	7.87	1040.09	165.29	476.11
13-6400571	39608	63.61	66	2842.77	7.87	3255.62	149.55	928.31
14-1513238	5009	14.14	17	96.09	7.87	432.97	94.45	117.40
15-0398550	5060	13.46	16	108.11	7.87	448.28	70.84	118.59
16-1251902	501	1.27	2	0.00	7.87	41.38	23.61	11.74
22-1211670	33471	64.10	71	2346.29	7.87	2914.82	181.04	784.48
22-1446668	269	0.73	0	0.00	7.87	23.34	7.87	6.30
22-2761297	5266	13.56	14	116.11	7.87	454.38	94.45	123.42
22-6001794	500	1.25	1	0.00	7.87	44.82	7.87	11.72
22-6001820	999	2.85	3	0.00	7.87	86.24	39.36	23.41
23-0671120	5034	12.88	13	96.09	7.87	429.92	102.32	117.98
23-1265004	1004	2.72	3	0.00	7.87	87.62	23.61	23.53
23-1943113	5010	12.76	14	92.09	7.87	425.77	94.45	117.42
23-1979193	501	1.44	2	0.00	7.87	44.09	23.61	11.74
23-1989084	30331	48.62	52	1317.29	7.87	1772.30	196.78	710.88
31-0559589	5051	13.11	15	108.11	7.87	439.53	78.71	118.38
31-1045381	500	1.32	2	0.00	7.87	43.77	23.61	11.72
31-1112315	250	0.44	1	0.00	7.87	14.64	7.87	5.86
31-1153510	39895	56.96	59	1749.71	7.87	2194.10	212.52	935.04
31-6000142	1002	2.56	2	0.00	7.87	83.82	23.61	23.48
33-0363116	7982	15.69	17	256.25	7.87	687.77	78.71	187.08
34-0505560	4028	9.98	11	68.07	7.87	340.72	70.84	94.41
34-0640780	501	1.37	2	0.00	7.87	43.02	23.61	11.74
34-0846262	3984	10.11	11	56.05	7.87	320.01	78.71	93.38
34-4431174	500	1.31	1	0.00	7.87	43.32	7.87	11.72
34-9990000	250001	440.02	464	29364.65	23.61	27814.53	94.45	5859.40
35-1152814	1005	2.80	3	0.00	7.87	89.61	23.61	23.55
35-6000169	997	2.36	3	0.00	7.87	87.20	23.61	23.37
36-1115800	40471	60.29	64	1793.75	7.87	2285.02	244.00	948.54
36-1282500	49059	87.08	95	3987.89	23.61	4247.56	196.78	1149.82
36-1750680	470474	752.20	888	46417.29	23.61	41123.82	228.26	11026.73
36-2179782	5050	13.14	15	104.10	7.87	439.52	94.45	118.36
36-2513909	2492	6.97	7	12.01	7.87	220.57	62.97	58.41
36-2708624	250	0.69	1	0.00	7.87	21.83	7.87	5.86
36-3540022	1334	3.83	4	0.00	7.87	113.59	39.36	31.27
38-0281180	250	0.65	1	0.00	7.87	22.10	7.87	5.86
38-1510762	375000	448.39	483	24948.34	23.61	21369.76	236.13	8789.06
38-1798424	49664	83.92	104	3927.83	7.87	4346.99	220.39	1164.00
38-2072341	1001	2.55	2	0.00	7.87	85.40	23.61	23.46
38-2113393	53558	78.39	83	3743.65	7.87	3909.66	188.91	1255.27
38-2706314	998	2.48	3	0.00	7.87	82.16	23.61	23.39
38-6001599	5058	13.50	14	120.12	7.87	477.28	78.71	118.55
38-6004706	250	0.56	1	0.00	7.87	20.89	7.87	5.86
38-6006309	51709	90.11	92	4364.26	23.61	4595.02	196.78	1211.93
39-0967678	5018	12.69	13	104.10	7.87	442.39	102.32	117.61
39-1390489	29	0.09	1	0.00	7.87	2.61	7.87	0.68
41-0749934	20174	39.67	41	1053.03	7.87	1607.38	181.04	472.83
41-0760000	434578	518.10	537	29785.06	23.61	25072.81	236.13	10185.42
41-0901437	41426	71.92	82	3147.07	7.87	3628.33	196.78	970.92



41-1243894	997	3.02	4	0.00	7.87	84.43	23.61	23.37
42-6004813	30186	55.38	58	2186.13	7.87	2615.60	220.39	707.48
43-1236588	250	0.56	0	0.00	7.87	20.89	7.87	5.86
44-0545813	997	2.74	3	0.00	7.87	87.97	31.48	23.37
44-6000524	2488	6.38	7	16.02	7.87	217.45	55.10	58.31
51-0331454	3973	10.30	11	100.10	7.87	369.93	70.84	93.12
52-1661226	250	0.63	1	0.00	7.87	21.54	7.87	5.86
52-6000989	20166	39.91	40	1281.25	7.87	1769.02	165.29	472.64
52-6002033	123843	184.16	188	11347.07	23.61	10765.24	196.78	2902.57
53-0196580	3971	10.35	10	72.07	7.87	357.50	70.84	93.07
53-0204616	1003	2.67	2	0.00	7.87	56.86	39.36	23.51
53-9990000	281634	331.90	620	18425.98	23.61	16525.01	220.39	6600.80
54-0332635	4011	9.81	16	56.05	7.87	331.48	70.84	94.01
54-1549763	1002	2.73	3	0.00	7.87	81.95	31.48	23.48
54-6001605	997	2.69	3	0.00	7.87	87.69	23.61	23.37
54-6001718	4009	10.53	26	36.04	7.87	324.10	78.71	93.96
55-0548701	501	1.13	2	0.00	7.87	44.20	7.87	11.74
56-1336585	501	1.41	2	0.00	7.87	42.81	23.61	11.74
57-0736794	499	1.39	2	0.00	7.87	39.82	23.61	11.70
57-0874699	501	1.33	2	0.00	7.87	43.09	23.61	11.74
58-0964286	3995	9.04	22	20.02	7.87	220.23	70.84	93.63
58-1516994	2494	6.68	13	12.01	7.87	218.46	62.97	58.45
58-6000147	500	1.36	2	0.00	7.87	39.91	23.61	11.72
59-2594590	594	1.69	4	0.00	7.87	49.80	23.61	13.92
61-1078924	499	1.34	1	0.00	7.87	43.43	23.61	11.70
61-1158928	250	0.56	1	0.00	7.87	21.29	7.87	5.86
62-0721803	3983	9.08	21	28.03	7.87	227.19	70.84	93.35
62-1091294	500	1.31	2	0.00	7.87	42.02	23.61	11.72
63-0570350	250	0.64	2	0.00	7.87	20.95	7.87	5.86
63-0581231	1229	3.32	4	0.00	7.87	108.34	39.36	28.80
63-0941966	39669	68.27	175	2870.80	23.61	3452.04	188.91	929.74
63-1032906	2499	6.61	13	12.01	7.87	209.55	70.84	58.57
64-0752022	500	1.29	2	0.00	7.87	41.55	23.61	11.72
68-0038644	1002	2.69	4	0.00	7.87	82.74	31.48	23.48
71-0225020	4010	10.20	25	56.05	7.87	353.20	70.84	93.98
71-0415188	267590	306.87	861	17204.79	23.61	14861.16	196.78	6271.64
71-0517471	3971	10.16	19	88.09	7.87	367.09	70.84	93.07
71-6021158	4029	10.07	17	68.07	7.87	318.23	70.84	94.43
72-0992142	500	1.30	2	0.00	7.87	41.03	23.61	11.72
73-1032203	2493	7.02	10	12.01	7.87	218.69	62.97	58.43
74-1109741	5012	12.65	23	108.11	7.87	409.29	86.58	117.47
74-1659656	4009	6.43	9	24.02	7.87	229.56	94.45	93.96
74-2248983	30024	46.05	95	1157.13	7.87	1679.41	196.78	703.69
74-2355451	250	0.66	2	0.00	7.87	22.22	7.87	5.86
75-1232789	123565	190.20	210	11022.75	23.61	10816.05	212.52	2896.05
75-2255014	3996	10.28	11	48.05	7.87	339.41	70.84	93.66
76-0100695	500	1.47	1	0.00	7.87	40.13	23.61	11.72
76-0146568	1003	2.95	3	0.00	7.87	85.80	39.36	23.51
76-0178498	20235	40.71	41	1277.25	7.87	1802.88	196.78	474.26
76-0264097	5049	12.56	13	96.09	7.87	404.36	86.58	118.34
76-0276305	501	1.50	2	0.00	7.87	46.46	23.61	11.74
84-0377058	4029	10.71	11	60.06	7.87	350.72	70.84	94.43
84-0997508	250	0.65	1	0.00	7.87	19.69	7.87	5.86
84-6010331	998	2.84	3	0.00	7.87	85.66	23.61	23.39

84-9980000	205891	307.37	314	19222.75	23.61	18227.09	236.13	4825.57
84-9990000	344999	526.97	538	34253.42	23.61	30639.63	244.00	8085.91
86-0096778	500	1.45	2	0.00	7.87	41.73	23.61	11.72
86-6000547	2497	6.75	7	12.01	7.87	221.70	55.10	58.52
87-6000525	40070	67.43	69	3071.00	7.87	3508.05	204.65	939.14
91-0907451	250	0.60	1	0.00	7.87	22.03	7.87	5.86
91-1043642	2500	6.99	7	16.02	7.87	213.20	62.97	58.59
91-1484586	250	0.67	1	0.00	7.87	22.95	7.87	5.86
93-0432081	48327	78.91	80	3719.63	7.87	4124.88	204.65	1132.66
93-0765253	499	1.38	1	0.00	7.87	40.67	23.61	11.70
94-0902780	501	1.41	2	0.00	7.87	44.76	23.61	11.74
94-1461843	2509	7.17	8	12.01	7.87	220.12	70.84	58.80
94-1606174	2490	7.06	7	8.01	7.87	205.59	70.84	58.36
94-1648752	6603	16.90	17	220.21	7.87	605.58	118.07	154.76
94-1675108	499	1.47	2	0.00	7.87	43.83	23.61	11.70
94-2131571	250	0.55	1	0.00	7.87	21.88	7.87	5.86
94-2629822	271324	392.89	401	24551.95	23.61	22035.67	228.26	6359.16
94-3041767	19921	34.26	36	728.71	7.87	1146.16	181.04	466.90
94-3070458	250	0.78	1	0.00	7.87	21.88	7.87	5.86
95-1699855	501	1.42	2	0.00	7.87	42.53	23.61	11.74
95-1978576	9923	23.49	25	372.36	7.87	861.16	133.81	232.57
95-3537532	4028	10.50	11	52.05	7.87	342.34	70.84	94.41
95-3800369	499	1.42	1	0.00	7.87	43.34	23.61	11.70
95-4172255	250	0.56	1	0.00	7.87	22.28	7.87	5.86
95-6000827	500	1.49	2	0.00	7.87	43.61	23.61	11.72
95-6042622	1003	3.15	3	0.00	7.87	88.40	31.48	23.51
03-6000658	501	1.40	1	0.00	7.87	45.28	23.61	11.74
04-1734655	10117	23.98	25	472.46	7.87	907.37	133.81	237.12
04-2156078	250	0.60	1	0.00	7.87	21.41	7.87	5.86
06-1204645	250	0.63	0	0.00	7.87	21.05	7.87	5.86
11-1001790	1000	2.99	3	0.00	7.87	86.34	31.48	23.44
11-2834450	250	0.70	1	0.00	7.87	21.55	7.87	5.86
13-3106295	4974	12.03	12	48.05	7.87	284.68	78.71	116.58
13-3173586	500	1.71	2	0.00	7.87	43.55	23.61	11.72
13-5318100	76672	122.24	127	6414.26	23.61	6736.77	244.00	1797.00
14-1588290	250	0.62	1	0.00	7.87	21.14	7.87	5.86
14-1682942	2499	6.83	7	12.01	7.87	220.52	62.97	58.57
15-0543659	2499	7.01	8	12.01	7.87	219.72	70.84	58.57
16-0743084	500	1.34	1	0.00	7.87	41.09	23.61	11.72
21-0635010	1002	2.81	3	0.00	7.87	88.81	31.48	23.48
22-1801227	1000	2.79	3	0.00	7.87	85.12	31.48	23.44
22-2341770	1001	3.00	4	0.00	7.87	88.64	39.36	23.46
23-1352620	1000	2.92	4	0.00	7.87	88.36	31.48	23.44
23-1668435	500	1.26	2	0.00	7.87	44.46	7.87	11.72
23-1691403	500	1.31	2	0.00	7.87	43.51	7.87	11.72
23-2578152	999	2.47	3	0.00	7.87	54.02	31.48	23.41
25-1211482	1001	2.78	3	0.00	7.87	89.31	23.61	23.46
25-1581273	1002	2.46	3	0.00	7.87	88.19	23.61	23.48
25-6000141	1002	2.66	3	0.00	7.87	88.45	23.61	23.48
31-0859857	1002	2.80	3	0.00	7.87	88.32	23.61	23.48
33-0195457	250	0.73	1	0.00	7.87	20.52	7.87	5.86
34-1580269	15037	25.32	25	504.49	7.87	862.28	133.81	352.43
34-4428218	997	2.67	3	0.00	7.87	86.26	23.61	23.37
35-1797634	500	1.33	1	0.00	7.87	42.47	7.87	11.72



35-9990000	658215	1097.77	2079	68506.84	23.61	60539.40	228.26	15426.91
36-1717960	5065	13.92	45	92.09	7.87	440.48	102.32	118.71
36-2590063	752	2.43	3	0.00	7.87	64.62	23.61	17.62
36-2930831	501	1.41	2	0.00	7.87	44.53	23.61	11.74
36-3429602	500	1.45	1	0.00	7.87	43.09	23.61	11.72
36-3515372	500	1.43	1	0.00	7.87	40.05	23.61	11.72
37-1174785	250	0.82	1	0.00	7.87	21.54	7.87	5.86
38-0763360	250	0.69	0	0.00	7.87	20.85	7.87	5.86
38-1274536	77981	123.90	127	6954.79	23.61	6818.62	220.39	1827.68
38-1369604	1004	2.78	3	0.00	7.87	86.22	31.48	23.53
38-1709248	9949	23.45	25	384.38	7.87	866.02	118.07	233.18
38-2500514	996	2.71	3	0.00	7.87	81.69	23.61	23.34
38-6006063	2499	6.60	7	16.02	7.87	221.11	39.36	58.57
38-6029206	4990	12.88	13	120.12	7.87	440.69	102.32	116.95
39-0379990	4025	10.87	11	64.06	7.87	356.57	86.58	94.34
39-1143171	250	0.64	1	0.00	7.87	22.15	7.87	5.86
41-0719940	1002	2.97	3	0.00	7.87	87.84	31.48	23.48
41-0848441	165398	265.10	271	15190.82	23.61	14349.83	212.52	3876.52
41-1260605	10046	23.36	25	416.41	7.87	847.15	133.81	235.45
41-1317059	2502	6.70	7	16.02	7.87	202.07	55.10	58.64
41-6034144	500	1.42	2	0.00	7.87	44.49	7.87	11.72
43-0977042	1001	2.86	3	0.00	7.87	86.23	31.48	23.46
47-0524851	1003	2.79	3	0.00	7.87	89.25	23.61	23.51
52-0886787	4020	10.88	11	60.06	7.87	352.12	78.71	94.22
53-0196597	500	1.58	2	0.00	7.87	45.52	23.61	11.72
54-6001128	4026	11.28	12	84.08	7.87	393.02	78.71	94.36
54-6001471	500	1.50	2	0.00	7.87	48.15	23.61	11.72
54-6001803	1000	2.71	3	0.00	7.87	57.79	39.36	23.44
56-0348490	3973	10.38	10	56.05	7.87	351.11	70.84	93.12
56-0792028	999	2.90	6	0.00	7.87	86.11	31.48	23.41
56-0940643	3980	10.93	12	56.05	7.87	344.27	70.84	93.28
56-1349827	555	1.50	2	0.00	7.87	47.49	23.61	13.01
56-1381211	2499	7.51	9	12.01	7.87	211.95	55.10	58.57
56-1703216	250	0.69	1	0.00	7.87	20.00	7.87	5.86
58-0978843	500	1.49	1	0.00	7.87	43.23	23.61	11.72
58-1837116	250	0.69	1	0.00	7.87	21.54	7.87	5.86
58-6000865	250	0.68	1	0.00	7.87	23.35	7.87	5.86
59-0248365	4943	13.31	14	108.11	7.87	428.42	94.45	115.85
59-0751858	250	0.87	1	0.00	7.87	22.56	7.87	5.86
59-0836811	1001	2.97	4	0.00	7.87	85.23	31.48	23.46
59-1672120	2509	7.93	13	12.01	7.87	215.32	70.84	58.80
59-1698006	13974	29.08	30	688.67	7.87	1209.01	133.81	327.52
59-2239528	5018	13.29	14	120.12	7.87	444.37	102.32	117.61
59-2612918	500	1.45	2	0.00	7.87	42.05	23.61	11.72
59-2964349	250	0.69	1	0.00	7.87	23.03	7.87	5.86
59-6000396	4023	11.05	11	56.05	7.87	353.27	70.84	94.29
61-0144470	5068	13.25	13	100.10	7.87	437.90	86.58	118.78
61-0592866	250	0.65	1	0.00	7.87	22.85	7.87	5.86
61-0997092	998	2.76	2	0.00	7.87	83.77	23.61	23.39
61-1028725	4981	12.74	14	108.11	7.87	431.04	102.32	116.74
62-0400610	2501	7.00	9	16.02	7.87	216.93	55.10	58.62
62-0450611	5067	13.93	18	116.11	7.87	445.18	102.32	118.76
62-1178938	998	2.93	4	0.00	7.87	93.17	39.36	23.39
63-0479282	3996	10.27	11	48.05	7.87	337.97	70.84	93.66



63-0857290	10035	23.01	24	400.39	7.87	837.77	133.81	235.20
63-0967763	250	0.62	0	0.00	7.87	22.60	7.87	5.86
63-6000997	4952	12.80	13	104.10	7.87	433.07	70.84	116.06
63-6001138	10114	23.79	24	436.43	7.87	873.18	133.81	237.05
65-0012091	996	2.81	3	0.00	7.87	85.95	31.48	23.34
65-0079270	500	1.44	2	0.00	7.87	43.44	23.61	11.72
71-0333344	250	0.70	1	0.00	7.87	22.39	7.87	5.86
71-0427007	79861	100.49	103	4432.32	7.87	4577.27	181.04	1871.74
71-0480157	1001	2.74	3	0.00	7.87	86.06	31.48	23.46
72-0741707	250	0.62	1	0.00	7.87	20.97	7.87	5.86
73-0579283	2494	6.89	7	12.01	7.87	206.65	62.97	58.45
74-0484030	10137	21.46	22	212.21	7.87	651.71	118.07	237.59
74-1109737	4027	10.96	12	52.05	7.87	350.79	70.84	94.38
74-1586031	5067	13.66	14	108.11	7.87	446.34	102.32	118.76
74-1750451	5017	12.38	13	72.07	7.87	367.98	70.84	117.59
74-2404626	999	2.63	3	0.00	7.87	87.38	23.61	23.41
74-2545137	250	0.72	1	0.00	7.87	22.14	7.87	5.86
75-0755367	1002	2.83	3	0.00	7.87	85.62	23.61	23.48
75-2257592	1002	3.10	3	0.00	7.87	85.61	39.36	23.48
76-0300290	501	1.41	2	0.00	7.87	44.62	23.61	11.74
77-0098061	4953	13.01	14	88.09	7.87	416.48	86.58	116.09
82-0325203	250	0.67	1	0.00	7.87	23.01	7.87	5.86
84-0602726	250	0.75	1	0.00	7.87	22.43	7.87	5.86
85-6005550	2500	6.95	7	20.02	7.87	208.89	70.84	58.59
86-0410573	250	0.72	1	0.00	7.87	19.96	7.87	5.86
86-0414274	10017	23.28	24	372.36	7.87	842.30	133.81	234.77
87-0368169	501	1.38	2	0.00	7.87	41.73	23.61	11.74
87-0453008	2494	7.01	8	16.02	7.87	209.30	62.97	58.45
88-0173471	250	0.69	1	0.00	7.87	20.71	7.87	5.86
91-0984002	501	1.71	2	0.00	7.87	44.59	23.61	11.74
93-0654045	250	0.67	1	0.00	7.87	19.32	7.87	5.86
94-0472650	2505	6.97	8	12.01	7.87	208.16	62.97	58.71
94-2290265	250	0.74	1	0.00	7.87	21.38	7.87	5.86
94-2507766	250	0.77	1	0.00	7.87	21.50	7.87	5.86
95-0613650	19317	38.22	39	964.94	7.87	1503.23	173.16	452.74
95-0831590	250	0.75	1	0.00	7.87	21.27	7.87	5.86
95-2013660	250	0.68	1	0.00	7.87	21.52	7.87	5.86
95-3946299	383	1.19	1	0.00	7.87	32.13	7.87	8.98
98-0018947	9979	23.92	25	588.57	7.87	932.97	118.07	233.88







